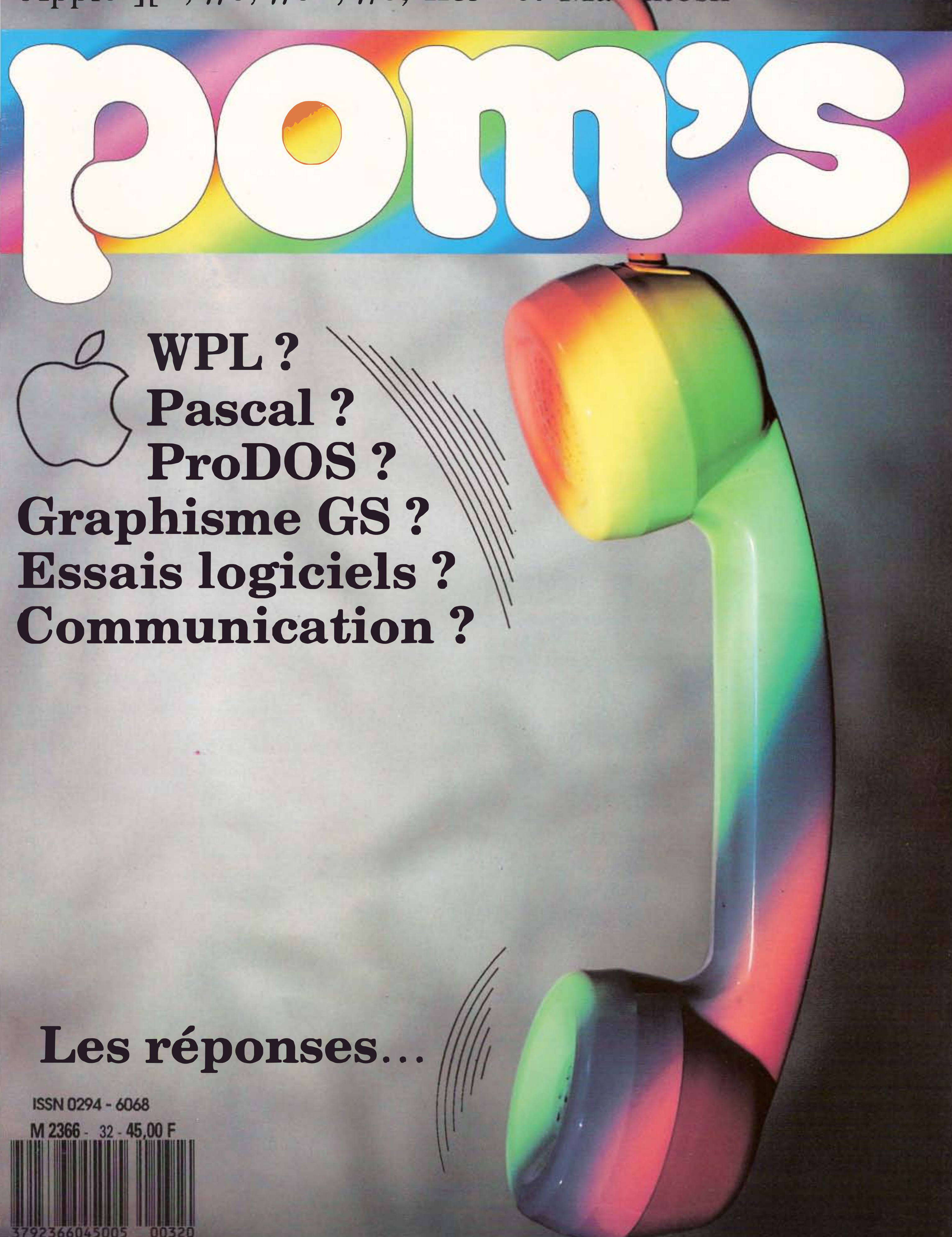
La revue francophone indépendante pour les utilisateurs des
Apple ][+, //e, //e+, //c, IIGS™ et Macintosh™



# Abonné à CalvaCom ?

Clv_Pom's, application "Qualité Pom's" pour le Macintosh, est un logiciel complet pour l'optimisation de vos connexions au serveur de RCI

Enregistrement des consultations dans des fichiers de type MacWrite

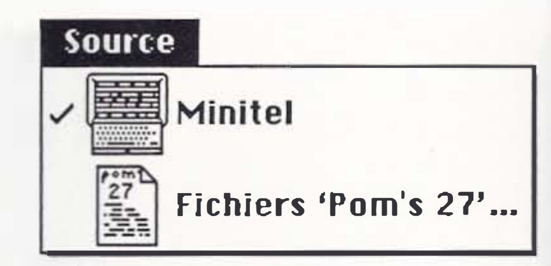
Exportation de messages, de procédures, de programmes et de fichiers

**Mode**
- ✓Enregistrement de textes
- Enregistrement de programmes
- Exportation de textes & procédures
- Exportation de programmes

**Fichier**
- Nouveau… ⌘N
- Ouvrir… ⌘O
- Suppression des lignes '--'
- ✓ Recodage é, à, etc.
- Impression simultanée…
- Fermer ⌘F

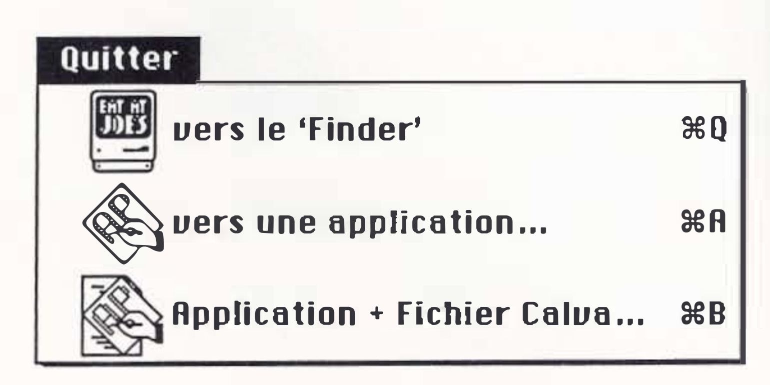
Suppression éventuelle des lignes de message du serveur

Option téléscripteur (impression des dépêches de l'AFP lorsqu'elles 'tombent' par exemple)




Possibilité de traitement des fichiers créés avec le programme 'Minitel 27'

**Téléchargement de programmes et fichiers**




Raffinement : en quittant Clv_Pom's, on peut se diriger vers le bureau électronique, vers une autre application mais aussi en 'emportant' le dernier fichier reçu…

200,00 F TTC franco, bon de commande page 74.
CLV_Pom's existe également en version Apple // pour utilisation avec un Minitel

## …Clv_Pom's

Numéro 32
septembre-octobre 1987





Éditions MEV - 12, rue d'Anjou - 78000 Versailles. Tél. : (1) 39 51 24 43. Directeur de la publication : Hervé Thiriez

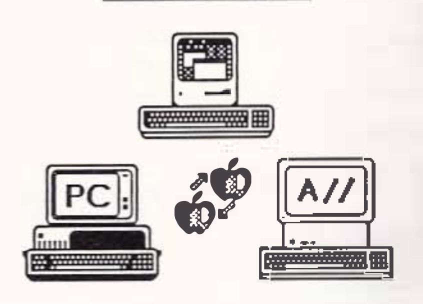
**Pour *TOUT* communiquer entre :**

**Apple //**
**Macintosh**
**IBM ou compatible**

---

# INTERPOM'S V 2.0

---













**Présentation dans le numéro 31 de Pom's**

**Ont collaboré à ce numéro**

Alexandre Avrane – Jean-Luc Bazanegue
Alain Bohec – Jean-Louis Chauvin
Paul Courbis – Robert Coustal
Alexandre Duback – Sylvie Gallet
Jean-Michel Gourévitch – Olivier Herz
Roland Jost – Philippe Mathieu
Gérard Michel – Christian Piard
Joelle Piard – Frédéric Rosay
H. Roy Contancin – Hervé Thiriez
Bernard Toméno – Vincent Toméno

**Directeur de la publication rédacteur en chef**

Hervé Thiriez

**Rédacteurs**

Alexandre Avrane – Olivier Herz

**Siège social**

Éditions MEV
12, rue d'Anjou
78000 Versailles
☎ (1) 39.51.24.43

**Publicité**

Éditions MEV

**Diffusion**

N.M.P.P.

**Impression**

Berger-Levrault
18, rue des Glacis
54000 Nancy
☎ 83.35.61.44

**Photo de couverture**

CP & JLB

**Photogravure**

Graphotec
21, chemin de la Tour
92350 Le Plessis-Robinson
☎ (1) 46.30.44.49



# Editorial

Grand rendez-vous annuel des Apple-maniaques, Apple-Expo vous propose cette année d'entrer gratuitement si vous êtes porteur de Pom's, bonne nouvelle.

Plus qu'une simple *Expo*, cette fête devrait nous réserver quelques grandes nouveautés : les HyperCard, MultiFinder et autre ImageWriter LQ mais on nous promet également la démonstration d'un réseau utilisant des fibres optiques, de multiples périphériques pour les handicapés (tablette graphique, écran tactile...) et le résultat de mille œuvres de développeurs. Bien entendu, Pom's vous y accueillera.

Ce numéro ? Toujours en évolution, nous avons ouvert Pom's à l'assembleur ORCA/M – GS oblige –, au Turbo Pascal sur Macintosh, et même à une grille de mots croisés, objet d'un petit concours. Autre nouveauté, votre courrier électronique : nombreux sont nos lecteurs qui, 24 heures sur 24, déposent dans notre boîte à lettres CalvaCom questions, remarques et suggestions ; deux pages de ce numéro en sont le reflet.

Pom's se veut une revue complète et variée et le pari est encore gagné pour cette rentrée : trois bancs d'essai, deux commandes externes ProDOS, un étonnant écran 'virtuel' en DOS 3.3, deux programmes Pascal, un petit montage électronique pour détecter la sonnerie du téléphone, un programme WPL... Deux mois entre chaque parution, ce n'est pas si long pour tout exploiter !

**Hervé Thiriez**

# Transformation d'images HGR en SHGR

Notre propos est d'apporter ici une solution pour transformer des images haute résolution (images classiques sur la gamme Apple ][, ][+ , //e, //c) en images de type GSPaint. Les systèmes de codage de ces deux types d'images sont très différents.

## Images classiques

- la résolution d'une image HGR classique est de 280 points horizontaux sur 192 points verticaux ;
- un point écran est codé sur 1 bit (allumé - éteint) ;
- un octet sert à coder 7 points (sur les 7 bits de plus faible poids, le bit de poids fort servant au codage de la couleur) . Une ligne écran est donc composée de 40 octets (40 octets * 7 bits = 280 points) ;
- l'organisation de la page HGR est assez compliquée car les adresses des premiers octets de chaque ligne ne sont pas en ordre régulièrement croissant (ainsi le 41ème octet au lieu d'être au début de la deuxième ligne est au début de la 64ème, le 81ème au début de la 128ème ligne ; tout se passe comme si l'écran était divisé en 3 zones de 64 lignes : pour plus de détails voir dans le 1er numéro de Pom's l'article sur l'organisation des pages graphiques) ;
- quoi qu'il en soit on peut obtenir l'adresse d'un point par la routine Moniteur HPOSN ($F411) après avoir chargé comme suit les registres :
  X & Y : position horizontale du point (faible-fort)
  A : position verticale
  le résultat est obtenu en lisant aux adresses $26 et $27.

## Images de type GSPaint

Le mode de construction d'une image de type GSPaint est plus simple :

- la résolution d'une telle image est de 320 points horizontaux sur 200 points verticaux ;
- un point écran est codé sur 4 bits (16 couleurs = 2 ^ 4 combinaisons de 4 bits) ;
- un octet sert donc à coder 2 points (et on a 160 octets sur une ligne) ;
- l'adressage est on ne peut plus simple : les 160 premiers octets pour la 1ère ligne, les 160 suivants pour la 2ème ligne, les 160 suivants pour la 3ème ligne, etc. ;
- un écran GSPaint se compose donc de 32000 octets (200 * 160) qui occupent la zone $E1/2000 à $E1/9CFF mais en fait le fichier sur disque en occupe 32768 car viennent ensuite les Scan Line Control Block et palettes (voir Pom's 31, Peeks et Pokes longs).

## Transformations d'images

La transformation d'une image classique en une image de type GSPaint, grâce au programme binaire que nous proposons s'effectue en plusieurs étapes :

- par manque de place en mémoire principale (Bank 00) pour y charger l'image HGR (4000 octets) et y reconstruire l'image type GSPaint (32768 octets), tout en gardant de la place pour les programmes, on transfère l'image à transformer en Bank 01 (mémoire auxiliaire de la carte 80 colonnes) grâce à AUX.MOVE ($C311) ;
- pour chaque ligne écran, on calcule l'adresse de ses 7 premiers points (c'est à dire du premier octet) ;
- l'octet qui s'y trouve ainsi que le suivant sont décomposés en leurs 14 bits significatifs ;
- puis on reproduit 4 fois chaque bit , ce qui nous donne 4 * 14 = 56 bits , que l'on recombine 8 par 8 pour faire 7 octets, qui sont alors stockés, en mémoire principale à partir de $1000, les uns à la suite des autres ;
- on passe ensuite aux 14 points suivants, puis à la ligne suivante, en incrémentant à chaque fois l'adresse de stockage des 7 octets ;
- une palette standard de couleurs est créée en $8E00.

Enfin, grâce à la routine MOVE que nous avons ajouté au programme binaire, le tout est déplacé en Bank E1 à partir de l'adresse $2000 pour pouvoir être visualisé.

Les fichiers qu'utilisent GSPaint sont de deux types :

- type $C0, format compacté ,
- type $C1, format écran.

Ce dernier format a été retenu et c'est pour pouvoir sauver sous cette forme qu'il faut au préalable créer un fichier de type $C1 par l'ordre Basic :

```
CREATE FICH, T$C1
```

avant de sauvegarder par :

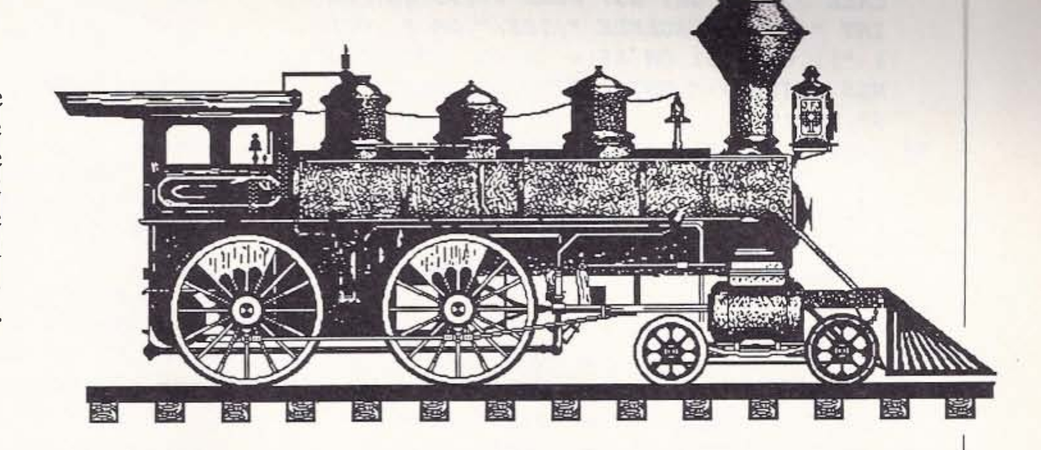
```
BSAVE FICH, A$1000, L32768
```

Tout cela est effectué par le programme Basic TRANSFORMEUR après qu'il vous ait demandé le nom de l'image à charger et qu'il vous ait proposé d'en respecter ou d'en bouleverser les couleurs.

# Vincent & Bernard TOMENO

À noter que l'image finale n'occupe pas toute la largeur de l'écran GSPaint (marge droite de 40 points et marge inférieure de 8 points) en raison de la différence entre les 320/200 points de la page SHGR du IIGS et les 280/192 points de l'image HGR).





## Récapitulation HGR.SUPHGR.C

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par :*
*BSAVE HGR.SUPHGR.C, A$9000,L$230*

```
9000:A9 00 85 F9 85 FD 85 3C
9008:85 3E 85 42 A9 20 85 3D
9010:85 43 A9 40 85 3F A9 10
9018:85 FE 38 20 11 C3 A9 00
9020:85 FA 20 DF 91 A0 00 AF
9028:00 00 01 20 67 91 AD 28
9030:90 8D 90 90 8D DD 90 EE
9038:90 90 EE DD 90 EE DD 90
9040:AD 29 90 8D 91 90 8D DE
9048:90 A9 02 85 CE A5 18 85
9050:CF A2 01 A5 1F 85 07 A9
9058:01 20 27 91 85 EE E8 A9
9060:00 20 27 91 85 EF 20 C7
9068:91 E8 A9 01 20 27 91 85
9070:EE E8 A9 00 20 27 91 85
9078:EF 20 C7 91 E8 A9 01 20
9080:27 91 85 EE E8 A9 00 20
9088:27 91 85 EF 20 C7 91 AF
9090:00 00 01 20 67 91 A2 00
9098:A9 01 20 27 91 85 EE E8
90A0:A5 1F 85 07 A9 00 20 27
90A8:91 85 EF 20 C7 91 E8 A9
90B0:01 20 27 91 85 EE E8 A9
90B8:00 20 27 91 85 EF 20 C7
90C0:91 E8 A9 01 20 27 91 85
90C8:EE E8 A9 00 20 27 91 85
90D0:EF 20 C7 91 E8 A9 01 20
90D8:27 91 85 EE AF 00 00 01
90E0:20 67 91 A2 00 A9 00 20
90E8:27 91 85 EF 20 C7 91 EE
90F0:DD 90 EE DD 90 EE 90 90
90F8:EE 90 90 AD DD 90 D0 06
9100:EE DE 90 EE 91 90 E6 FA
9108:A5 FA C9 14 F0 03 4C 51
9110:90 18 A5 FD 69 14 85 FD
9118:D0 02 E6 FE E6 F9 A5 F9
9120:C9 C0 F0 15 4C 1E 90 85
9128:08 A5 CE 85 09 A5 CF 85
9130:CE B5 18 85 CF 20 79 91
9138:60 A2 00 BD 47 91 9D 00
9140:8E E8 E0 20 D0 F5 60 00
9148:00 77 07 41 08 2C 07 0F
9150:00 80 00 70 0F 00 0D A9
9158:0F F0 0F E0 00 DF 04 AF
9160:0D 8F 07 CC 0C FF 0F 85
9168:FF A2 00 46 FF A9 00 69
9170:00 95 18 E8 E0 08 D0 F3
9178:60 A5 CE C9 00 D0 22 A5
9180:09 C9 00 F0 3F A5 CF C9
9188:00 F0 39 A5 08 C9 01 F0
9190:08 A5 07 C9 00 F0 2A D0
9198:25 A5 07 C9 00 F0 1C D0
91A0:17 A5 09 C9 01 F0 0E A5
91A8:CF C9 01 F0 08 A5 08 C9
91B0:01 F0 DE D0 E4 A9 0F 60
91B8:A9 06 60 A9 0A 60 A9 04
91C0:60 A9 03 60 A9 00 60 A5
91C8:EE 0A 0A 0A 0A 85 06 18
91D0:A5 EF 65 06 91 FD E6 FD
91D8:A5 FD D0 02 E6 FE 60 A5
91E0:F9 A0 00 A2 00 20 11 F4
91E8:A5 26 8D 28 90 A5 27 8D
91F0:29 90 60 A9 00 85 D6 A9
91F8:80 85 D7 A2 00 A0 00 AF
9200:00 10 00 8F 00 20 E1 EE
9208:04 92 EE 00 92 C8 C4 D6
9210:D0 ED EE 05 92 EE 01 92
9218:E8 E4 D7 D0 E0 A9 00 8D
9220:04 92 8D 00 92 A9 20 8D
9228:05 92 A9 10 8D 01 92 60
```

## Programme TRANSFORMEUR

```
10 D$ =  CHR$ (4): PRINT D$"BLOAD HGR.SU
   PHGR.C": PRINT : PRINT D$"PR£3": PRIN
   T : ONERR  GOTO 125
15  HOME : TEXT : VTAB 2: HTAB 3: INVERS
   E : PRINT "TRANSFORMATION D'IMAGE CLA
   SSIQUE HGR EN IMAGE DE TYPE GS/PAINT
   POUR II GS"
20  NORMAL : CALL  - 958: PRINT : HTAB 2
   8: PRINT "(NE MARCHE QUE SUR II GS)":
    FOR I = 1 TO 3: POKE 49204,I + 4: PO
   KE 49186,I + 12: FOR D = 1 TO 300: NE
   XT : NEXT : REM  POKES POUR FAIRE JOL
   I
25  VTAB 8: INPUT "              NOM DE L
   'IMAGE A TRANSFORMER ( RTN = CATALOG
    ) ";IT$: IF IT$ =  CHR$ (3) THEN 125
30  IF IT$ = "" THEN  HOME : PRINT : PRI
   NT D$"CAT": GET A$: HOME : GOTO 25
35  HGR : PRINT D$"BLOAD"IT$",A$2000":AD
    = 37302: POKE AD,15: POKE AD + 3,6:
   POKE AD + 6,10: POKE AD + 9,4: POKE A
   D + 12,3: POKE AD + 15,0
40  VTAB 22: PRINT " RESPECT DES COULEUR
   S ? (O/N) ";: GET A$: IF A$ = "N" THE
   N 65
45  HOME : VTAB 22: PRINT " OK POUR TRAN
   SFORMATION ? (O/N) ";: GET A$: IF A$
```

```
< > "O" THEN  TEXT : HOME : GOTO 25
50  CALL 36864: PRINT : POKE 49193,161:
    CALL 37363: GET A$: POKE 49193,65: PR
    INT " ON SAUVEGARDE ";IT$;".GS ? (O/N
    ) ";: GET A$: ON A$ < > "O" GOTO 15:
    NS$ = IT$ + ".GS": PRINT D$"CREATE"NS
    $",T$C1"
55  PRINT D$"BSAVE"NS$",T$C1,A$1000,L327
    68": HOME : TEXT : VTAB 15: HTAB 3: I
    NVERSE : PRINT NS$;" EST MAINTENANT A
    U CATALOG": PRINT : HTAB 8: PRINT "ET
     LISIBLE SOUS GS/PAINT": NORMAL
60  END
65  PRINT :T = 1
70  IF T = 1 THEN CL$ = "NOIR":IN = 15
75  IF T = 2 THEN CL$ = "VIOLET":IN = 12
80  IF T = 3 THEN CL$ = "BLEU":IN = 9
85  IF T = 4 THEN CL$ = "VERT":IN = 6
90  IF T = 5 THEN CL$ = "ORANGE":IN = 3
95  IF T = 6 THEN CL$ = "BLANC":IN = 0
100  IF T > 6 THEN  HOME : GOTO 45
105  VTAB 22: CALL  - 958: HTAB 12: PRIN
    T " QUELLE NUANCE POUR TRADUIRE LE ";
    CL$;" ( RTN = NO CHANGE )"
110  PRINT "   0 = Noir , 1 = Gris foncé
     , 2 = Brun , 3 = Violet , 4= Bleu fo
    ncé , 5 = Vert foncé , 6 = Orange , 7
     = Rouge , 9 = Jaune , 10 = Vert clai
    r , 11 = Bleu clair , 12 = Mauve , 13
     = Bleu moyen , 14 = Gris clair , 15
    = Blanc";
115  VTAB 21: HTAB 72: INPUT CO$:T = T +
     1: ON CO$ = "" GOTO 70:CO =  VAL (CO
    $): IF CO > = 0 OR CO < 16 THEN  POK
    E AD + IN,CO
120  GOTO 70
125 ER =  PEEK (222):LI =  PEEK (218) +
     PEEK (219) * 256:ER$ = "Y a un probl
    ème en ligne " +  STR$ (LI): IF ER =
    255 THEN  HOME : END
130  IF ER = 6 OR ER = 7 THEN ER$ = "Err
    eur sur le nom du Fichier ou du Volum
    e"
135  IF ER = 4 THEN ER$ = "Disque protég
    é en écriture"
140  IF ER = 3 OR ER = 8 THEN ER$ = "Err
    eur d'entrée/sortie"
145  IF ER = 9 THEN ER$ = "Pas assez de
    place sur ce disque": PRINT D$"DELETE
    "NS$
150  IF ER = 19 THEN ER$ = "Fichier déja
     existant sur le disque"
155  HOME : VTAB 23: HTAB  INT ((80 -  L
    EN (ER$)) / 2): PRINT ER$; CHR$ (7):
    FOR D = 1 TO 3000: NEXT : HOME : TEXT
     : GOTO 25
160  REM  LES POKES AD (LIGNES 35 ET 11
    5) RETABLISSENT OU CHANGENT LES COULE
    URS ; CALL 36864 = ADRESSE DU PGM BIN
    ., CALL 37363  =  ROUTINE METTANT L'I
    MAGE EN $E1/2000
```

## Source HGR.SUPHGR.S
### Assembleur ProCODE

ProDOS

]|gs

```
********************************
*------------------------------*
*!  TRANSFORMATION D'IMAGES   !*
*!   HGR -> SHGR  format GS   !*
*! Tomsoftware      JUIN 1987 !*
*!  ASSEMBLEUR    PROCODE     !*
*!  ( BIG MAC Compatible )    !*
*------------------------------*
********************************

PY        =    $F9     ; numéro de ligne
PX        =    $FA     ; numéro d'octet dans la ligne/2
LO.DS     =    $FD
HI.DS     =    $FE
FIRST     =    $EE
SECND     =    $EF
CASE      =    $FF     ; Qq adresses de stockage...
OCTET     =    $6      ; ...des données...
BIT       =    $18     ; ...pendant l'opération
COUL      =    $7      ; BIT codant la couleur
RANG      =    $8      ; colonne paire/impaire
BIT1      =    $9      ; pixel d'avant
BIT2      =    $CE     ; pixel à l'étude
BIT3      =    $CF     ; pixel d'après
MP.OL     =    $3C     ; Origine de la ....
MP.OH     =    $3D     ;...zone à transferrer
MP.FL     =    $3E     ; Fin de la zone...
MP.FH     =    $3F     ;... à transferrer
MA.OL     =    $42     ; Origine de la ....
MA.OH     =    $43     ;...zone destination
AUX.MOVE  =    $C311
PALETTE   =    $8E00
HPOSN     =    $F411
GBASL     =    $26
GBASH     =    $27

*

***************
*  PROGRAMME  *
***************
*

          ORG  $9000

* INITIALISATION DES ADRESSES *
*_____________________________*

          LDA  £$00
          STA  PY
          STA  LO.DS
          STA  MP.OL
          STA  MP.FL
          STA  MA.OL
          LDA  £$20
          STA  MP.OH
          STA  MA.OH
          LDA  £$40
          STA  MP.FH
          LDA  £$10          ; L'image GS sera stockée à
          STA  HI.DS         ; partir de $1000 en Mem. princ.

* TRANSFERT IMAGE EN MEM.AUX. *
*_____________________________*

          SEC                ; Retenue à 1 pour transfert de
          JSR  AUX.MOVE      ; la Mem Princ.-->Mem Aux.

* BOUCLE SUR L'IMAGE *
```

```
*_____________________*

DEBUT   LDA  £0
        STA  PX
        JSR  ADRESSE


        LDY  £500
        HEX  AF          ; OPCODE DE LDA LONG
ADIM    HEX  0000        ; = LO ET HI DE L'OCTET A ETUDIER
        HEX  01          ; DANS LA BANK 01 ( = MEM.AUX.)
        JSR  LABO
        LDA  ADIM
        STA  ADIM2
        STA  ADIM1
        INC  ADIM2
        INC  ADIM1
        INC  ADIM1
        LDA  ADIM+1
        STA  ADIM2+1
        STA  ADIM1+1

* RECONSTRUCTION DES 7 OCTETS *

        LDA  £02         ; pour respecter la couleur du
        STA  BIT2        ; 1° pixel de chaque ligne
        LDA  BIT
        STA  BIT3

DEBSEC  LDX  £01
        LDA  BIT+7
        STA  COUL
        LDA  £01
        JSR  DECAL
        STA  FIRST       ; 1° pixel d'un octet GS/PAINT
        INX              ; X=2
        LDA  £00
        JSR  DECAL
        STA  SECND       ; 2° pixel de l'octet GS/PAINT
        JSR  RECONS

        INX              ; X=3
        LDA  £01
        JSR  DECAL
        STA  FIRST
        INX              ; X=4
        LDA  £00
        JSR  DECAL
        STA  SECND
        JSR  RECONS

        INX              ; X=5
        LDA  £01
        JSR  DECAL
        STA  FIRST
        INX              ; X=6
        LDA  £00
        JSR  DECAL
        STA  SECND
        JSR  RECONS


        HEX  AF          ; OPCODE DE LDA LONG
ADIM2   HEX  0000        ; 2eme OCTET A DECORTIQUER
        HEX  01          ; EN BANK 01
        JSR  LABO

        LDX  £00
        LDA  £01
        JSR  DECAL
        STA  FIRST
        INX              ; X=1
        LDA  BIT+7
        STA  COUL
        LDA  £00
        JSR  DECAL
```

```
        STA  SECNO
        JSR  RECONS

        INX              ; X=2
        LDA  £01
        JSR  DECAL
        STA  FIRST
        INX              ; X=3
        LDA  £00
        JSR  DECAL
        STA  SECND
        JSR  RECONS

        INX              ; X=4
        LDA  £01
        JSR  DECAL
        STA  FIRST
        INX              ; X=5
        LDA  £00
        JSR  DECAL
        STA  SECNO
        JSR  RECONS

        INX              ; X=6
        LDA  £01
        JSR  DECAL
        STA  FIRST

        HEX  AF          ; OPCODE DE LDA LONG
ADIM1   HEX  0000        ; 1er OCTET A DECORTIQUER
        HEX  01          ; EN BANK 01
        JSR  LABO

        LDX  £00
        LDA  £00
        JSR  DECAL
        STA  SECNO
        JSR  RECONS

        INC  ADIMI
        INC  ADIM1
        INC  ADIM2
        INC  ADIM2
        LDA  ADIM1
        BNE  NINC
        INC  ADIM1+1
        INC  ADIM2+1
NINC    INC  PX
        LDA  PX
        CMP  £20
        BEO  NEXT
        JMP  OEBSEC      ; on passe aux autres points
NEXT    CLC
        LDA  LO.DS
        AOC  £20
        STA  LO.DS
        BNE  NO.INC
        INC  HI.DS
NO.INC  INC  PY          ; incrementation ligne.
        LDA  PY
        CMP  £192        ; derniere ligne ?
        BEQ  CREPAL      ; si oui, creation palette.
        JMP  DEBUT       ; sinon, ligne suivante.

* DECALAGE VERS LE PIXEL SUIVANT *
*_____________________________*

DECAL   STA  RANG
        LDA  BIT2        ;
        STA  BITI        ;
        LDA  BIT3        ; pixel suivant
        STA  BIT2        ;
        LDA  BIT,X       ;
        STA  BIT3        ;
        JSR  CRECOUL
        RTS

* CREATION DE LA PALETTE DE COULEURS *
*__________________________________*
```

```
CREPAL   LDX  £00          ; Creation palette GS.PAINT
         LDA  TABLE,X      ; La palette standard
         STA  PALETTE,X    ; est composée
         INX               ; des 32 octets de la table
         CPX  £32          ; ci dessous à raison de
         BNE  CREPAL+2     ; 2 octets par couleur
         RTS

TABLE    HEX  0000770741082C07
         HEX  0F008000700F000D
         HEX  A90FF00FE000DF04
         HEX  AF0D8F07CC0CFF0F

* LABORATOIRE *
*____________*

* DISSECTION D'UN OCTET *

LABO     STA  CASE         ; l'octet est rangé dans une case
                           ; de travail
         LDX  £00
ROT      LSR  CASE         ;BIT de droite va dans la retenue
         LDA  £00
         ADC  £00          ; donc Accumulateur = retenue
         STA  BIT,X        ; le BIT est mis de coté
         INX
         CPX  £08          ; dernier BIT ?
         BNE  ROT

         RTS

* CALCUL DE LA COULEUR D'UN PIXEL HGR *
*____________________________________*

CRECOUL  LDA  BIT2         ; Bit du pixel en etude
         CMP  £00
         BNE  EGUN
         LDA  BIT1         ; Bit du pixel precedent
         CMP  £00
         BEQ  NOIR
         LDA  BIT3         ; Bit du pixel suivant
         CMP  £00
         BEQ  NOIR
         LDA  RANG
         CMP  £01
         BEQ  IMPAIR
PAIR     LDA  COUL
         CMP  £00
         BEQ  VIOLET
         BNE  BLEU
IMPAIR   LDA  COUL
         CMP  £00
         BEQ  VERT
         BNE  ORANGE
EGUN     LDA  BIT1         ; Bit du pixel precedent
         CMP  £01
         BEQ  BLANC
         LDA  BIT3         ; Bit du pixel suivant
         CMP  £01
         BEQ  BLANC
         LDA  RANG
         CMP  £01
         BEQ  PAIR
         BNE  IMPAIR

BLANC    LDA  £15          ; les 16 couleurs GS/PAINT sont
         RTS               ; codées comme suit :
ORANGE   LDA  £06          ; NOIR = 0  GRIS FONCE = 1
         RTS               ; MARRON = 2  VIOLET = 3
VERT     LDA  £10          ; BLEU FONCE = 4  VERT FONCE = 5
         RTS               ; ORANGE = 6  ROUGE = 7  ROSE = 8
BLEU     LDA  £04          ; JAUNE = 9  VERT CLAIR = 10
         RTS               ; BLEU CLAIR = 11  MAUVE = 12
VIOLET   LDA  £03          ; BLEU MOY. = 13  GRIS CLAIR = 14
         RTS               ; BLANC = 15
NOIR     LDA  £00          ; Dans les LDA ci-contre vous
         RTS               ; pouvez remplacer 6 par 7,10 par
                           ; 5 , etc... voire des valeurs de
                           ; couleurs totalement différentes
```

```
* RECONSTRUCTION D'UN OCTET GS/PAINT *
*___________________________________*

RECONS   LDA  FIRST        ; premier pixel de l'octet
         ASL
         ASL
         ASL
         ASL
         STA  OCTET
         CLC
         LDA  SECND        ; second pixel de l'octet
         ADC  OCTET
         STA  (LO.DS),Y    ; (HI.DS LO.DS): adresse oct
         INC  LO.DS        ; GS/PAINT reconstitué.
         LDA  LO.DS
         BNE  RETOUR
         INC  HI.DS
RETOUR   RTS

* CALCUL DE L'ADRESSE DU PREMIER OCTET D'UNE LIGNE HGR
*______________________________________________________

ADRESSE  LDA  PY
         LDY  £00
         LDX  £00
         JSR  HPOSN
         LDA  GBASL
         STA  ADIM
         LDA  GBASH
         STA  ADIM+1
         RTS

****************************************
* ROUTINE MOVE POUR TRANSFERER L'IMAGE *
*     DE $00/1000 VERS $E1/2000        *
****************************************

COMP1    =    $D6          ;COMPTEUR 256 OCTETS
COMP2    =    $D7          ;COMPTEUR 128*256

MOVE     LDA  £00
         STA  COMP1
         LDA  £128
         STA  COMP2
         LDX  £00
TOUR1    LDY  £00

SOURCE   HEX  AF001000     ; LDA EN BANK 00

DESTIN   HEX  8F0020E1     ; STA EN BANK E1
         INC  DESTIN+1
         INC  SOURCE+1
         INY
         CPY  COMP1        ;256 OCTETS ?
         BNE  TOUR1+2
         INC  DESTIN+2
         INC  SOURCE+2
         INX
         CPX  COMP2        ;128° BLOC DE 256 ?
         BNE  TOUR1

         LDA  £00          ;ON RETABLIT...
         STA  DESTIN+1     ;les adresses SOURCE et ...
         STA  SOURCE+1     ;DESTIN comme elles ...
         LDA  £S20         ;étaient...
         STA  DESTIN+2     ;au départ...
         LDA  £$10         ;...
         STA  SOURCE+2     ;...
         RTS

**********************************************
*            V. et B. TOMENO                 *
**********************************************
```

# Courbes fractales

## Sylvie Gallet

En guise de suite au programme du n° 22 de Pom's (dessins de courbes fractales sur Macintosh) et pour que l'Apple // ne soit pas en reste, voici un programme qui profite pleinement de la récursivité du Pascal pour aller un peu plus loin dans les espaces non entiers.

Le programme `Récursives` propose huit options :

### Courbe de Von-Koch

La courbe fractale la plus connue. (Se reporter au Pom's 22 pour sa construction)

### Triangle

Une autre fractale dans laquelle chaque segment :

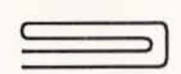
est remplacé par :



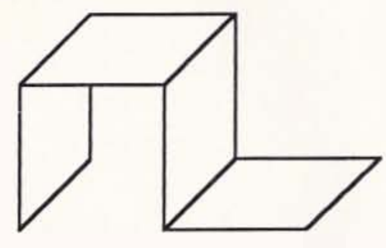
en changeant à chaque fois le sens de rotation.

### Dragon

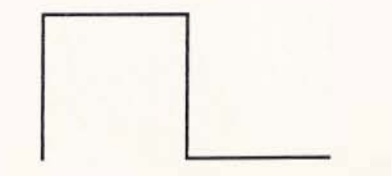
Cette courbe illustre ce que l'on obtient en pliant une bande de papier un certain nombre de fois (2 par exemple) :



et en la dépliant de façon que chaque pliure forme un angle droit :



ce qui donne la courbe :

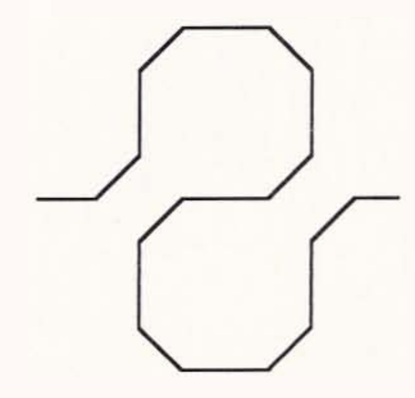


A noter que les ordres proposés pour cette courbe vont de 1 à 6 mais que l'ordre réel va de 2 à 12 par pas de 2.

### Courbe de Péano

Le principe est le suivant : un segment :

est remplacé par :



et chaque petit segment obtenu subit le même traitement. On obtient une courbe fermée en appliquant cette méthode à un carré et la courbe obtenue tend à remplir un autre carré.
Hélas, la haute résolution de l'Apple // est bien vite prise en défaut.

### La courbe de Sierpinski

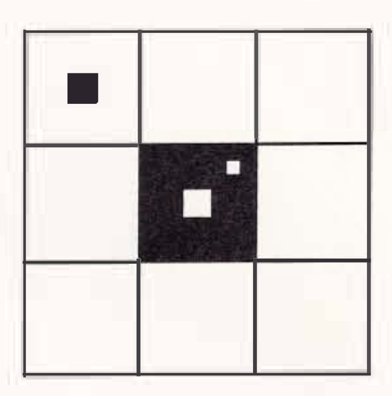
(traduite d'un programme en langage C paru dans Micro-Systèmes) : elle utilise 4 procédures mutuellement récursives pour calculer les coordonnées de chaque sommet.

### Hexagone

Un hexagone qui contient des hexagones qui contiennent des hexagones...

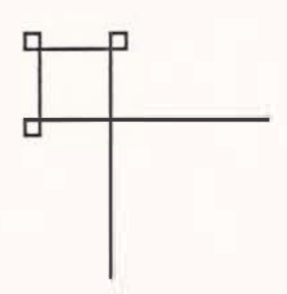
### Dentelle

On partage un carré en 9, on inverse la couleur du centre et on recommence sur chaque petit carré...



### Carrés

Un carré dont les sommets sont prolongés par des carrés plus petits dont les sommets...



### Impression

Si vous disposez d'un programme de dump sur imprimante, il vous suffit de le placer en librairie. Dans le cas contraire, toutes les instructions concernant l'impression doivent être supprimées du source.

### Sur les disquettes Pom's...

...les fichiers sont transférables sur votre disquette Pascal

- depuis la face DOS grâce à l'utilitaire Basic-Pascal livré avec ;
- depuis la face ProDOS grâce à l'utilitaire Universal File Conversion édité par Quality Software.



# Programme RECURSIVE.TEXT

```
(*$C       Sylvie GALLET              *)
(*$S+*)

program courbes_recursives ;

  uses turtlegraphics, dump_hgr ; (* si vous avez un programme de dump hgr pour
                                     votre imprimante, il est temps de le mettre
                                     en librairie! Sinon, il faut supprimet tout
                                     ce qui concerne cette instruction *)

  const marge = 20 ; (* 20 pour un ecran 80 colonnes , 0  pour 40 colonnes *)
  type choideca = set of char ;
         courbe = record
                    nom, niv : string
                  end ;

  var liste : array [1..8] of courbe ;
      i : integer ;
      efl, cr, son, home, reponse : char ;
      papier : text ;

  (*                  procedures d'interet general                         *)

  procedure prenreturn ;
    var sort : char ;
    begin
      repeat
        read (keyboard, sort) ;
      until eoln (keyboard)
    end ;

  function prencar (bonset : choideca ) : char ;
    var ch : char ;
        bon : boolean ;
    begin
      repeat
        read (keyboard,ch) ;
        if eoln (keyboard) then ch := cr ;
        bon := ch in bonset ;
        if not bon then
          write (son)
        else
          if ch in [' '..'z'] then
            write (ch)
      until bon ;
      prencar := ch
    end ;

  procedure message (x, y : integer ; s : string ) ;
    begin
      gotoxy (x,y) ; write (s, efl)
    end ;
```

][+
//e
//e+
//c
][gs

```
(*        affiche le menu pour chaque courbe et retourne le choix          *)

function menu ( choix : string ) : char ;
  var bonset : choideca ;
  begin
    if choix = 'D' then
      begin
        message (marge,6,'D - dessin ') ; bonset := ['D','d']
      end
    else
      begin
        bonset := ['1'..choix[length(choix)]] ;
        message (marge,6, concat ('ordre : ',choix ))
      end ;
    message (marge,8, 'I - imprimer ') ;
    message (marge,10, 'Q - quitter ') ;
    message (marge+15,12, 'votre choix : ') ;
    bonset := bonset + ['I','Q','i','q'] ;
    menu := prencar (bonset)
  end ;

(*          cette procedure contient toutes les procedures de dessin       *)

procedure dessine_courbe ;
  var reponse : char ;
      ordre : integer ;

  procedure init_hgr (x, y, angle : integer) ;

    begin
      initturtle ; pencolor (none) ; moveto (x,y) ;
      turnto (angle) ; pencolor (white)
    end ;

  procedure von_koch ;

    procedure v_dessin (ordre, long : integer) ;
      var nouv_long, nouv_ordre : integer ;
      begin
        nouv_ordre := ordre-1 ;
        nouv_long := long div 3 ;
        if ordre = 0 then
          move (long)
        else
          begin
            v_dessin ( nouv_ordre, nouv_long) ;
            turn (60) ;   v_dessin ( nouv_ordre, nouv_long) ;
            turn (-120) ; v_dessin ( nouv_ordre, nouv_long) ;
            turn (60) ;   v_dessin ( nouv_ordre, nouv_long)
          end
      end ;

    begin
      init_hgr (0,40,0) ; v_dessin (ordre, 270) ;
      readln
```

```
    end ;

(*--------------------------------------------------------------------------*)

  procedure triangle ;
    var long : integer ;

    procedure t_dessin (ordre, long, sens : integer ) ;
      var nouv_long, nouv_ordre : integer ;
      begin
        nouv_ordre:= ordre-1 ;
        nouv_long:= long div 2 ;
        if ordre = 0 then
          move (long)
        else
          begin
            turn ( 60*sens) ; t_dessin (nouv_ordre, nouv_long, -1*sens) ;
            turn (-60*sens) ; t_dessin (nouv_ordre, nouv_long, sens) ;
            turn (-60*sens) ; t_dessin (nouv_ordre, nouv_long, -1*sens) ;
            turn ( 60*sens)
          end
      end ;

    begin
      case ordre of
        1,2,6 : long := 256 ;
        3,5   : long := 255 ;
        4     : long := 239
      end ;
      init_hgr (12,0,0) ; t_dessin (ordre, LONG,1) ;
      readln
    end ;

(*--------------------------------------------------------------------------*)

  procedure dragon ;

    procedure d_dessin (ordre, sens : integer ; long : real) ;
      var nouv_ordre, nouv_sens : integer ;
          nouv_long : real ;
    begin
      nouv_ordre := ordre - 1 ;
      nouv_sens := sens * (-1) ;
      nouv_long := long * 0.70711 ;
      if ordre = 0 then
        move (trunc(long))
      else
        begin
          turn ( 45*sens) ; d_dessin (nouv_ordre, 1, nouv_long ) ;
          turn (-90*sens) ; d_dessin (nouv_ordre,-1, nouv_long ) ;
          turn ( 45*sens)
        end
    end ;

    begin
      init_hgr (69,53,0) ; ordre := 2 * ordre ;
       d_dessin (ordre, 1, 160) ; readln
    end ;

(*--------------------------------------------------------------------------*)

  procedure peano ;
    var long, i : integer ;

    procedure p_dessin (i, long, sens : integer ) ;
      var long1, long2, long3 : integer ;
      begin
        if i > 1 then
          begin
            long3 := long div 3 ;
            p_dessin (i-1, long3, -1) ; turn (-90) ;
            p_dessin (i-1, long3,  1) ; turn ( 90) ;
            p_dessin (i-1, long3,  1) ; turn ( 90) ;
            p_dessin (i-1, long3,  1) ; turn ( 90) ;
            p_dessin (i-1, long3, -1) ; turn (-90) ;
            p_dessin (i-1, long3, -1) ; turn (-90) ;
            p_dessin (i-1, long3, -1) ; turn (-90) ;
            p_dessin (i-1, long3,  1) ; turn ( 90) ;
            p_dessin (i-1, long3, sens)
          end
        else
          begin
            long1 := long div 2 ;
            long2 := trunc (long * 0.35355) ;
            move (long1) ; turn (-45) ; move (long2) ; turn (-45) ;
            move (long1) ; turn ( 45) ; move (long2) ; turn ( 45) ;
            move (long1) ; turn ( 45) ; move (long2) ; turn ( 45) ;
            move (long1) ; turn ( 45) ; move (long2) ; turn ( 45) ;
            move (long1) ; turn (-45) ; move (long2) ; turn (-45) ;
            move (long1) ; turn (-45) ; move (long2) ; turn (-45) ;
            move (long1) ; turn (-45) ; move (long2) ; turn (-45) ;
            move (long1) ; turn ( 45) ; move (long2) ; turn ( 45) ;
            move (long1) ; turn ( 45*sens) ; move (long2) ;
            turn (-45*sens) ;
          end ;
      end;

    begin
      init_hgr (135,0,-45) ;
      if ordre = 3 then
        long := 56
      else
        long := 48 ;
      for i := 1 to 4 do
        begin
          turn (90) ; p_dessin(ordre,long,1)
        end ;
      readln
    end ;

(*--------------------------------------------------------------------------*)
```

```
procedure sierpinski ;
  var h, i, z, x, y, n : integer ;

  procedure cote_1 ( i : integer ) ; forward ;

  procedure cote_2 ( i : integer ) ; forward ;

  procedure cote_3 ( i : integer ) ; forward ;

  procedure cote_4 ( i : integer ) ;
    begin
      if i > 0 then
        begin
          cote_4 (i-1) ; x := x + h ; y := y + h ;
          moveto (x,y) ; cote_1 (i-1) ;
          y := y + 2 * h ; moveto (x,y) ;
          cote_3 (i-1) ; x := x - h ; y := y + h ;
          moveto (x,y) ; cote_4 (i-1)
        end
    end ;

  procedure cote_3 ;
    begin
      if i > 0 then
        begin
          cote_3 (i-1) ; x := x - h ; y := y + h ;
          moveto (x,y) ; cote_4 (i-1) ;
          x := x - 2 * h ; moveto (x,y) ;
          cote_2 (i-1) ; x := x - h ; y := y - h ;
          moveto (x,y) ; cote_3 (i-1)
        end
    end ;

  procedure cote_2 ;
    begin
      if i > 0 then
        begin
          cote_2 (i-1) ; x := x - h ; y := y - h ;
          moveto (x,y) ; cote_3 (i-1) ;
          y := y - 2 * h ; moveto (x,y) ;
          cote_1 (i-1) ; x := x + h ; y := y - h ;
          moveto (x,y) ; cote_2 (i-1)
        end
    end ;

  procedure cote_1 ;
    begin
      if i > 0 then
        begin
          cote_1 (i-1) ; x := x + h ; y := y - h ;
          moveto (x,y) ; cote_2 (i-1) ;
          x := x + 2 * h ; moveto (x,y) ;
          cote_4 (i-1) ; x := x + h ; y := y + h ;
          moveto (x,y) ; cote_1 (i-1)
        end
    end ;

  begin
    i := 0 ; n := ordre ; h := 48 ;
    for z := 1 to n do
      begin
        h := h div 2 ; i := i + 1
      end ;
    x := 40 + h ; y := 191 - h ; init_hgr (x,y,0) ;
    cote_1 (i) ; x := x + h ; y := y - h ; moveto (x,y) ;
    cote_2 (i) ; x := x - h ; y := y - h ; moveto (x,y) ;
    cote_3 (i) ; x := x - h ; y := y + h ; moveto (x,y) ;
    cote_4 (i) ; x := x + h ; y := y + h ; moveto (x,y) ;
    readln
  end ;

(*---------------------------------------------------------------------------*)

procedure hexagone ;

  procedure h_dessin (long : integer) ;
    var cote, nouv_long : integer ;
    begin
      if long > 3 then
        begin
          nouv_long:= trunc (long*0.36) ;
          for cote := 1 to 6 do
            begin
              move (nouv_long) ; turn (60) ; h_dessin (nouv_long)
            end
        end
    end ;

  begin
    init_hgr (255,95,120) ; h_dessin (264) ;
    readln
  end ;

(*---------------------------------------------------------------------------*)

procedure dentelle ;

  procedure de_dessin (x, y, long : integer ) ;
    var i,j : integer ;
    begin
      if long > 0 then
        for i := 0 to 2 do
          for j := 0 to 2 do
            if i*j = 1 then
              begin
                viewport (x+long, x+2*long-1, y+long, y+2*long-1) ;
                fillscreen (reverse) ;
                de_dessin (x+long, y+long, long div 3)
              end
            else
```

```
            de_dessin (x+long*i, y+long*j, long div 3) ;
    end ;

  begin
    init_hgr (0,0,0) ; de_dessin (60,15,54) ;
    readln
  end ;

(*---------------------------------------------------------------------------*)


procedure carres ;
  var i : integer ;

  procedure c_dessin (long, sens : integer ) ;
    var i,nlong,nsens : integer ;
    begin
      if long > 1 then
        begin
          nlong := long div 2 ; nsens := -1*sens ;
          for i := 1 to 3 do
            begin
              move (long) ; c_dessin (nlong,nsens)
            end ;
          move (long)
        end
      else
        turn (90*sens)
    end ;

  begin
    init_hgr (100,64,0) ;
    for i := 1 to 4 do
      begin
        move (64) ; c_deesin (32,1)
      end ;
    readln
  end ;

(*---------------------------------------------------------------------------*)

 begin      (* dessine_courbe *)
  repeat
    textmode ;write (home) ;
    gotoxy (15,1) ; writeln (liste[i].nom) ;
    reponse := menu (liste[i].niv) ;
    if not (reponse in ['Q','q'] ) then
      if reponse in ['I','i'] then
        begin
          message (0,14, 'preparez l''imprimante puis tapez RETURN ') ;
          prenreturn ; rewrite (papier, 'printer:') ;
          write (papier , cr, liste [i].nom ) ;
          if i in [1..5] then
              writeln (papier, ',  ordre  ',ordre) ;
          dump ;
          writeln (papier, cr, '---------------------------------------') ;
          close (papier)
        end
      else
        begin
          ordre := ord (reponse)-ord ('0') ;
          case i of
            1 : von_koch ;
            2 : triangle ;
            3 : dragon ;
            4 : peano ;
            5 : sierpinski ;
            6 : hexagone ;
            7 : dentelle ;
            8 : carres
          end
        end ;
  until reponse in ['Q','q']
end;


(*---------------------------------------------------------------------------*)
(*                          menu principal                                   *)
(*---------------------------------------------------------------------------*)

begin
  home := chr (12) ; cr := chr (13) ; efl := chr (29) ; son := chr (7) ;
  liste [1].nom := '1 - courbe de Von Koch' ;       liste [1].niv := '1..4' ;
  liste [2].nom := '2 - triangle' ;                 liste [2].niv := '1..6' ;
  liste [3].nom := '3 - dragon' ;                   liste [3].niv := '1..6' ;
  liste [4].nom := '4 - courbe de Peano' ;          liste [4].niv := '1..3' ;
  liste [5].nom := '5 - courbe de Sierpinski' ;     liste [5].niv := '1..4' ;
  liste [6].nom := '6 - hexagone' ;                 liste [6].niv := 'D' ;
  liste [7].nom := '7 - dentelle' ;                 liste [7].niv := 'D' ;
  liste [8].nom := '8 - cerres' ;                   liste [8].niv := 'D' ;
  repeat
    write (home) ;
    message (marge,1,'Trace de courbes recursives') ;
    message (marge,2,'---------------------------') ;
    for i := 1 to 8 do
      message (marge,i*2+2, liste [i].nom ) ;


    message (marge,20, 'Q - quitter ') ;
    message (marge+15,23, 'votre choix :') ;
    reponse :=prencar (['1'..'8','Q','q']) ;
    if not (reponse in ['Q','q'] ) then
      begin
        i := ord (reponse) - ord ('0') ;
        dessine_courbe
      end ;
  until reponse in ['Q','q']
end.
```

# Kruptos

*Pour l'Apple //, une commande externe ProDOS,*
*Pour le Macintosh, un accessoire de bureau :*

*Uløiyu mKØÙlY^∞*
*ÒYMlioÍb¥§'21≈b^$)*
*à~æ∞0¨n∞ÒN∫·ÙÒÇpç!7¨ÚÔ∫√*

*Kruptos est un utilitaire de cryptage disponible à tout instant. Kruptos rend inaccessible sans la clef de décodage tous les fichiers qui doivent rester confidentiels : courrier personnel, rapport professionnel, fichier de clients. Même les programmes et applications peuvent être protégés.*

*Les deux versions de Kruptos sont des programmes Pom's, listés dans la revue n° 29.*

*Revue 29 : 45,00 F*

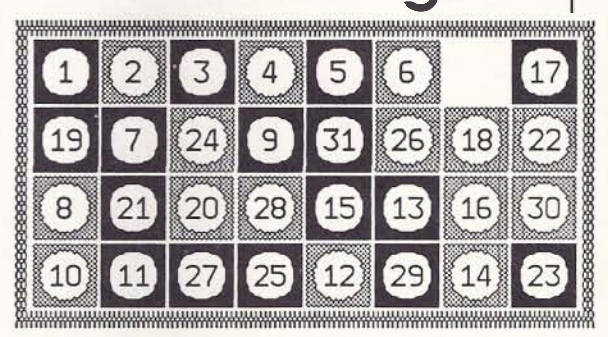
*Disquettes :*
*Apple // 140Ko : 60,00 F*
*Apple // 800Ko : 80,00 F*
*Macintosh : 80,00 F*

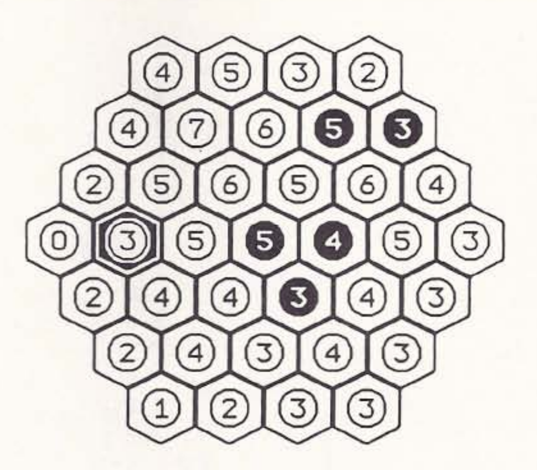
*Bon de commande page 74*

# Une disquette de jeux : Ludologic

Au sommaire de la disquette LUDOLOGIC, trois jeux de réflexion de difficulté croissante. Ces jeux qui nécessitent des neurones aussi calmes qu'entraînés, ne devraient pas décevoir les amateurs de puzzles et autres casse-têtes.

Il n'est pas nécessaire de présenter TAQUIN, ce pousse-pousse informatique ici fort bien présenté.
Nouvelle difficulté, NOIR & BLANC : 37 hexagones peuvent être noirs ou blancs mais au départ vous n'en connaissez pas la couleur. Chacun comporte un numéro qui représente le nombre de cellules voisines blanches... À vous de reconstituer le décor original !
HEXAGONE MAGIQUE est encore plus délicat, même principe que le carré magique, mais ici vous devrez installer les chiffres de 1 à 19 dans un hexagone de telle façon que les 5 horizontales et 10 obliques totalisent chacune 38 : bonne chance.




Fidèle à son habitude, Pom's vous propose sur cette disquette les sources des routines écrites par Sylvie Gallet en assembleur Lisa 2.5. Bien entendu, le Basic est également listable. TAQUIN et NOIR & BLANC utilisent leur propre routine graphique qui permet de dessiner plus rapidement qu'avec des shapes.

**80,00 F Franco,**
**Bon de commande page 74**

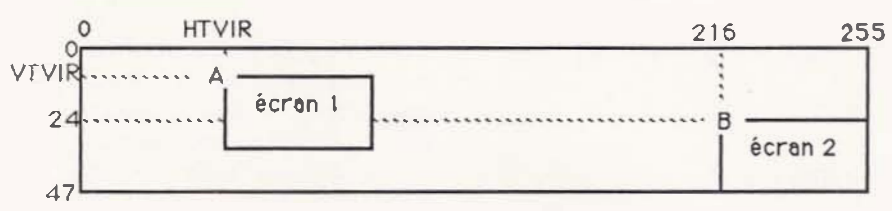
# Ecran virtuel

H. Roy-Contancin

Voici un programme qui permet de disposer d'un écran virtuel de 256 colonnes par 48 lignes de texte (ou 128 par 96). Bien entendu, il ne sera pas possible d'afficher l'ensemble de l'écran, mais votre écran physique de 40 colonnes et 24 lignes devient une fenêtre que vous déplacez dans les quatre directions grâce aux flèches du clavier.

Pour ce qui est de l'usage de ce programme, elle dépend de l'imagination : stockage d'écran (jusqu'à 15), menus défilants, écrans type Visicalc.

## Principe de l'écran virtuel

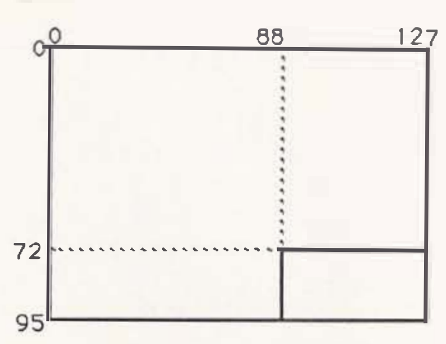



En 256 colonnes, les lignes sont numérotées de 0 à 47, la variable VTVIR ($07) représentant le numéro de ligne. Les colonnes sont numérotées de 0 à 255, la variable HTVIR représentant le numéro de la colonne.

L'écran 1 est localisé par les coordonnées du point A qui est le coin supérieur gauche de l'écran physique.

Le point B localise l'écran inférieur droit qui a les coordonnées maximales pour la fonction d'affichage.

L'écran 128 colonnes se présente ainsi :




## Les commandes

Les différents paramètres à transmettre sont des variables Basic ou des constantes (A, A%, A(25), A%(25)...).

Pour la présentation ci-dessous, les variables ont la signification suivante :

| | |
|---|---|
| LD | Ligne début |
| CD | Colonne début |
| VT | VTVIR |
| HT | HTVIR |
| LF | Ligne fin |
| CF | Colonne fin |
| L | Toutes les L lignes |
| C | Toutes les C colonnes |

### Home
### &H, LD, LF, CD, CF

Effacement de tout ou partie de l'écran virtuel. Les 4 paramètres sont obligatoires ; pour effacer tout l'écran, il faut :

256 col : &H, 0, 47, 0, 256

128 col : &H, 0, 95, 0, 127

La fonction effacement ne change pas l'affichage physique.

### Copy
### &C, VT, HT

Cette fonction copie l'écran physique de telle façon que le point VTAB 1 HTAB 1 soit placé en VT, HT sur l'écran virtuel. Le programme contrôle que VT et HT ne dépassent pas les coordonnées du point B.

### Window
### &W, VT, HT

Fonction inverse de la précédente : on place la fenêtre définie par VT, HT sur l'écran physique. Mêmes contrôles.

### Print
### &P, VT, HT

Imprime sur l'écran virtuel le contenu de la variable définie **en premier** dans le programme. Cette variable sert de buffer d'impression.

```
10 A$ = "": REM définition
20...
...
...
175 A$ = "Hello"
180 &P,VT,HT : REM Affiche
```

Ce PRINT n'a pas d'effet immédiat sur l'écran : &W, VT, HT le rend visible.

Le PRINT virtuel respecte les fonctions NORMAL, FLASH et INVERSE du Basic.

La variable de transfert (A$ dans l'exemple) doit être définie en premier dans le programme Basic (gare au D$ = CHR$(4) fréquemment défini en tête des programmes...).

### Grand cadre
### &K, 1

L'écran virtuel est bordé de blancs inversés, très utiles pour repérer les limites. Attention, les fonctions &C et &P écrasent sans complexe le cadre. &K, 1 rétablit la situation.

### Quadrillage
### &K, 2, L, C

Cette fonction trace le grand cadre et trace une ligne toutes les L lignes et une colonne toutes les C colonnes.

### Visualisation
### &V, VT, HT

Par cette commande, VTVIR et HTVIR sont mis à VT et HT et la fenêtre peut être dirigée à votre gré à l'aide des 4 flèches (sur le ][+, CTRL-K et CTRL-J

remplacent ↑ et ↓).
On sort de la visualisation par ESC ou RETURN. Si l'utilisateur a pressé ESC, PEEK (9) = 141, s'il a pressé RETURN, PEEK (9) = 155.

## Turn page
## &T, 1 &T, 2

Un écran virtuel peut en cacher un autre :
La carte langage est constituée de 2 *banks* de 4Ko et d'une partie commune de 8Ko.
En 256 colonnes, les *banks* 1 ou 2 contiennent 16 lignes et les 8Ko 32 lignes.
En 128 colonnes, les *banks* 1 ou 2 contiennent 32 lignes et les 8Ko 64 lignes.
&T, 1 et &T, 2 permettent de choisir le *bank* utilisé.

## Colonnage 128/256
## Poke 10,0

Affichage en 128 colonnes. Par défaut, 256 colonnes.

## Save page
## Call 37723

Pour sauvegarder un écran, il convient d'en transférer le contenu de la carte 16Ko en $5180 par CALL 37723 puis de faire BSAVE PAGE, A$5180, L$4000

## Load page
## Call 37742

Pour charger un écran, il convient de faire BLOAD PAGE puis d'en transférer le contenu en carte 16Ko par CALL 37742.

# VIRCEL, programme de démonstration

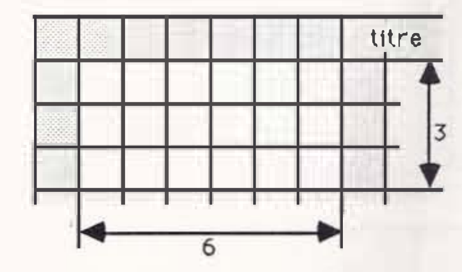
À titre de démonstration, le programme VIRCEL utilise la plupart des commandes de l'écran virtuel.

Il permet de créer et de tenir à jour un écran géant de 128 colonnes par 96 lignes, écran divisé en cellules pour tenir à jour un petit fichier d'adresses par exemple.

La cellule est un rectangle d'un nombre de lignes et de colonnes définis à la création de l'écran, ce sont des cases dans lesquelles on peut stocker du texte à volonté.

Une cellule est constituée d'une ligne titre (vidéo inverse) et de 1 à 20 lignes de texte.

Une ligne est constituée de 1 à 39 caractères, le premier caractère est en vidéo inverse et il est inaccessible.




## Le menu

### Chargement d'un écran

Le programme vous demande d'entrer le nom de l'écran voulu. Pour charger l'écran VIDEO, il suffit de taper VIDEO.

### Consultation

Utilisation directe de &V, VT, HT. On déplace l'écran par les flèches et on sort par ESC ou RETURN.

### Mise à jour

Il y a 3 phases à distinguer :

1 Visualisation de l'écran : vous devez jouer avec les flèches pour que la cellule à modifier soit présente à l'écran puis faire RETURN. ESC permet d'abandonner la mise à jour.
2 Affichage de * en vidéo inverse.
Déplacer l'* de façon à la loger dans la cellule à mettre à jour, puis faire RETURN (ou ESC pour abandonner).
Si vous faites sortir l'* de l'écran, on repasse en phase 1.
3 Saisie du texte dans la cellule. La cellule choisie est placée dans le coin supérieur gauche.

### Création d'un écran

Fonction très simple.

### Sauvegarde de l'écran

Le programme demande le nom de l'écran, ce qui permet de le dupliquer.

### *Note*

Ligne 1070 : les PEEK (33153) et 33154 servent à récupérer la taille d'une cellule (nombre de lignes et de colonnes). Ces informations ont été stockées à la création de l'écran (ligne 4175) dans les 2 premiers octets de la page 2, non utilisée par VIRCEL.

# Comment faire ?

Vous avez la disquette d'accompagnement Pom's, pas de problèmes, vous disposez des fichiers nécessaires. Pour voir la démonstration, faire simplement RUN VIRCEL.

Vous n'avez pas la disquette Pom's :

1 saisir et assembler BIGTEXT, BIGPGM et BIGSAVE pour obtenir les fichiers BIGTEXT.OBJ0, BIGPGM.OBJ0 et BIGSAVE.OBJ0,
2 saisir et sauvegarder la table BIGPTR,
3 saisir, sauvegarder et exécuter le programme VIRPUZZLE qui, à partir des 4 fichiers ci-dessus, constitue ECRVIRT.
4 pour vous servir de la démonstration, saisir, sauvegarder et exécuter VIRCEL.

---

# ✂ DhgrTool

Dans le numéro 30 de Pom's, un Couper/Coller énergique a fait perdre quelques octets à la récapitulation DHGRTOOL.0 ; la routine ne peut fonctionner qu'en ajoutant à partir de $0B96 :
84 20 2B EB 60

## Programme VIRPUZZLE

```
1  REM   VIRPUZZLE
5  REM :    RASSEMBLAGE DES MORCEAUX
10  HIMEM: 37248: REM  PROTECTION DU PRO
   GRAMME
20 D$ =  CHR$ (4)
30  PRINT D$"BRUN BIGTEXT.OBJ0"
40  PRINT D$"BLOAD BIGPTR,A$93D0"
45  PRINT D$"BLOAD BIGSAVE.OBJ0"
50  PRINT D$"BLOAD BIGPGM.OBJ0"
60  PRINT D$"BSAVE ECR-VIRT,A$9180,L$480
```

## Programme 'VIRCEL'

```
5  REM  GESTION CELLULES TEXTE
10  HIMEM: 20864: REM  PROTECTION ECRAN V
   IRTUEL
15 A$ = "A": REM  PREMIERE VARIABLE DU PG
   M
17 D$ =  CHR$ (4):B$ = "
      "
20  PRINT D$"BRUN ECR-VIRT": REM  CHARGEM
   ENT
30  POKE 10,0: REM  ECRAN 128 COLONNES
50  HOME : REM  AFFICHAGE MENU
60 : HTAB 10: INVERSE : PRINT "MENU": NOR
   MAL
70  VTAB 4: HTAB 1: PRINT "1 CHARGEMENT D
   'UN ECRAN"
80  VTAB 7: HTAB 1: PRINT "2 CONSULTATION
    CELLULES"
90  VTAB 10: HTAB 1: PRINT "3 MISE A JOUR
    CELLULES "
95  VTAB 13: HTAB 1: PRINT "4 CREATION D'
   UN ECRAN"
98  VTAB 16: HTAB 1: PRINT "5 SAUVEGARDE
   ECRAN"
100  VTAB 13: HTAB 1: PRINT "4 CREATION D
   'UN ECRAN  "
105  VTAB 23: GET E$: PRINT D$
110  IF E$ = "1" THEN  GOTO 1000
120  IF E$ = "2" THEN 2000
130  IF E$ = "3" THEN 3000
140  IF E$ = "4" THEN 4000
150  IF E$ = "5" THEN 5000
160  GOTO 50
1000  REM  CHARGEMENT ECRAN
1020  HOME : PRINT "NOM DE L'ECRAN ": INP
   UT E$
1030  IF  LEN (E$) = 0 THEN 1020
1040  ONERR  GOTO 1020
1045  PRINT "JE CHARGE"
1050  PRINT D$"BLOAD ";E$",A$5180"
1060  CALL 37742: REM  TRANSFERT SUR CART
   E 16K
1070 L =  PEEK (33153):Y =  PEEK (33154)
1080  GOTO 50
2000  REM  VISUALISATION
2005 VT = 0:HT = 0
2010  & V,VT,HT
2020  GOTO 50
3000  REM  MISE A JOUR
3005 VT = 0:HT = 0
3010  & V,VT,HT: REM  AFFICHAGE
3020 VT =  PEEK (7):HT =  PEEK (8)
3021  IF  PEEK (9) = 155 THEN 50: REM  ES
   CAPE
3025 V = 12:H = 20
3030  IF HT > 88 THEN HT = 88
3031  IF VT > 72 THEN VT = 72
3035  & W,VT,HT
3040  INVERSE : VTAB V: HTAB H: PRINT "*"
   : NORMAL
3050  GET E$: PRINT D$:A =  ASC (E$)
3060  IF A = 21 THEN 3100
3070  IF A = 8 THEN 3200
3080  IF A = 11 THEN 3300
3085  IF A = 10 THEN 3400
3086  IF A = 27 THEN 50: REM   ESCAPE
3088  IF A <  > 13 THEN 3030
3090  GOTO 3500
3100  REM  DROITE
3110  IF H = 40 THEN 3010
3120 H = H + 1: GOTO 3030
3200  IF H = 1 THEN 3010
3210 H = H - 1: GOTO 3030
3300  IF V = 1 THEN 3010
3310 V = V - 1: GOTO 3030
3400  IF V = 24 THEN 3010
3410 V = V + 1: GOTO 3030
3500  REM  MISE A JOUR EFFECTIVE
3510  HOME : REM  CALCUL DE V0,H0
3512 SY = Y + 1:H1 = HT + H - 1:V1 = VT +
    V - 1
3514 H0 = SY *  INT (H1 / SY) + 1:V0 = L
   *  INT (V1 / L)
3516  IF V0 + L > 95 THEN 3030
3518  IF H0 + SY > 128 THEN 3030
3520 D1 = 0:D2 = 0: REM  DEPLACEMENTS VER
   T/HORIZ
3530 V2 = V0:H2 = H0:
3540  IF V0 > 72 THEN V2 = 72:D1 = V0 - 7
   2
3550  IF H0 > 88 THEN H2 = 88:D2 = H0 - 8
   8
3560  & W,V2,H2 - 1: REM  AFFICHAGE CELLU
   LE DANS LE COIN SUPERIEUR
3590  VTAB (1 + D1): HTAB (1 + D2): INPUT
    T$: IF  LEN (T$) = 0 THEN 3030: REM
     TITRE
3600 A$ =  LEFT$ (T$ + B$,Y): INVERSE
3610  & P,V0,H0
3620  NORMAL : FOR J = 1 TO L - 1
3630  VTAB (J + 1 + D1): HTAB (1 + D2): I
   NPUT E$
3640 A$ =  LEFT$ (E$ + B$,Y): & P,V0 + J,
   H0
3650  NEXT J
3660  GOTO 3030
4000  REM  CREATION ECRAN
4030  HOME : HTAB 10: PRINT "CREATION D'U
```

DOS 3.3

][+
//e
//e+
//c
][gs

```
        N ECRAN"
4040  VTAB 5: HTAB 1: PRINT "NOMBRE DE LIGNES     "
4050  VTAB 8: HTAB 1: PRINT "NOMBRE DE COLONNES  "
4060  VTAB 11: HTAB 1: PRINT "NOM DE L'ECRAN"
4061  VTAB 5: HTAB 20: INPUT E$:X =  VAL (E$)
4065  IF X < 1 OR X > 20 THEN 4061
4070  VTAB 8: HTAB 20: INPUT E$:Y =  VAL (E$)
4075  IF Y < 1 OR Y > 39 THEN 4070
4080  VTAB 11: HTAB 20: INPUT E$: IF  LEN (E$) = 0 THEN 80
4090  & T,1: REM   PAGE 1
4100  POKE 10,0: REM   128 COLONNES
4110  & H,0,95,0,127: REM  HOME
4120 L = X + 1: & K,2,L,Y + 1
4130  INVERSE :A$ = E$: & P,0,1: NORMAL
4140  & W,0,0: REM MM AFFICHAGE RESULTAT
4145 F$ = E$
4150  VTAB 23: HTAB 1: PRINT "CONFIRMEZ PAR C"
4160  INPUT E$: IF E$ <  > "C" THEN 4030
4165  & W,0,0: VTAB 23: HTAB 1: PRINT "ATTENDEZ MERCI"
4170  CALL 37723: REM  TRANSFERT EN MEM
4175  POKE 33153,L: POKE 33154,Y
4180  PRINT D$"BSAVE ";F$;",A$5180,L$3010"
4190  GOTO 50
5000  REM  SAUVEGARDE
5010  PRINT "ENTREZ LE NOM DE L'ECRAN"
5020  INPUT E$: IF  LEN (E$) = 0 THEN 5005
5030  GOTO 4145
```

## Source 'BIGTEXT'

*Assembleur Big Mac, format TEXT*

```
*****************
* PAGE GEANTE
*
* H.ROY-CONTANCIN
* 19/05/86 1.1
****************
PAGE     EQU  $06
VTVIR    EQU  $07
HTVIR    EQU  $08
SAVX     EQU  $09
DIM1     EQU  $0A
DIM2     EQU  $0C
ADR      EQU  $18
ADP      EQU  $1A
RAM      EQU  $C083
ROM      EQU  $C082
LIG      EQU  $1C
LIGP     EQU  $1D
MAXV     EQU  $CE
MAXH     EQU  $CF
LONG     EQU  $1F
LIGD     EQU  $F9
LIGF     EQU  $FA
COLD     EQU  $FB
COLF     EQU  $FC
PTR      EQU  $FD
SAVA     EQU  $FF
GETBYTC  EQU  $E6F5
SPPRINT  EQU  $9400
SPCADRE  EQU  $9403
SPVISU   EQU  $9406
ADTX     EQU  $9409
INIADR   EQU  $940C
ERROR    EQU  $D419
***-CHARGEMENT-ADRESSE
         ORG  $9180
         LDA  #$4C
         STA  $3F5
         LDA  #$90
         STA  $3F6
         LDA  #$91
         STA  $3F7
         RTS
***-PRO-PRINCIPALE
         TAY
         LDA  DIM1
         BEQ  E128
E256     LDA  #$00
         STA  DIM1+1
         LDA  #$01
         STA  DIM1
         LDA  #$30
         STA  DIM2
         CLC
         BCC  NOPAG
E128     LDA  #$80
         STA  DIM1+1
         LDA  #$60
         STA  DIM2
NOPAG    LDA  PAGE
         CMP  #$02
         BEQ  PAG2
         LDA  #$00
         STA  SAVX
         CLC
         BCC  PPTEST
PAG2     LDA  #$08
         STA  SAVX
PPTEST   TYA
         CMP  #$48
         BEQ  BH
         CMP  #$43
         BEQ  BC
         CMP  #$57
         BEQ  BW
         CMP  #$50
         BEQ  BP
         CMP  #$4B
         BEQ  BK
         CMP  #$54
         BEQ  BT
         CMP  #$56
         BEQ  BV
         LDX  #$10
         JMP  ERROR
BH       JMP  SPHOME
BC       JMP  SPCOPY
BW       JMP  SPWIN
BP       JMP  SPPRINT
BK       JMP  SPCADRE
BT       JMP  SPCHPAG
BV       JMP  SPVISU
**************
*HOME
SPHOME   JSR  ADTX
         JSR  GETBYTC
         STX  LIGD
         TXA
         CMP  DIM2
         BCS  ANOH
         JSR  GETBYTC
         CPX  DIM2
         BCS  ANOH
         INX
         STX  LIGF
         TXA
         CMP  LIGD
         BCC  ANOH
         JSR  GETBYTC
         STX  COLD
         LDA  DIM1+1
         BEQ  GB4
         CPX  DIM1+1
         BCS  ANOH
GB4      JSR  GETBYTC
         STX  COLF
         LDA  DIM1+1
         BEQ  SPHS
         CPX  DIM1+1
         BCS  ANOH
SPHS     CPX  COLD
```

```
         BCC  ANOH
         BCS  SUITHOM
ANOH     LDX  #$4D
         JMP  ERROR
SUITHOM  JSR  INIADR
         LDX  #$00
BCL1     CPX  LIGD
         BCC  HSUIT
         CPX  LIGF
         BCC  HEFF
FIN      LDA  ROM
         RTS
HSUIT    INX
         JSR  AD80
         CLC
         BCC  BCL1
HEFF     LDY  COLD
         LDA  #$A0
HEF2     STA  (ADR),Y
         CPY  COLF
         INY
         BCC  HEF2
         BCS  HSUIT
********
AD80     CLC
         LDA  ADR
         ADC  DIM1+1
         STA  ADR
         LDA  DIM1
         ADC  ADR+1
         STA  ADR+1
         RTS
INPUT    JSR  ADTX
         JSR  GETBYTC
         STX  VTVIR
         CPX  MAXV
         BCS  SPCERR
         JSR  GETBYTC
         STX  HTVIR
         CPX  MAXH
         BCS  SPCERR
         LDA  #$00
         STA  LIGP
         STA  LIG
         STA  ADR
         LDA  #$D0
         STA  ADR+1
         LDX  SAVX
         LDA  RAM,X
         LDA  RAM,X
         LDA  #$D0
         STA  PTR
         LDA  #$93
         STA  PTR+1
         RTS
***** CACUL ADP
CALCADP  LDA  LIGP
         CLC
         ASL  A
         TAY
         LDA  (PTR),Y
         STA  ADP
         INY
         LDA  (PTR),Y
         STA  ADP+1
         RTS
SPCOPY   JSR  CHMAX
         JSR  INPUT
SPCB1    LDA  LIG
         CMP  VTVIR
         BCC  SPCLS
         CLC
         LDA  VTVIR
         ADC  #$17
         CMP  LIG
         BCC  SPCFIN
* ON STOCKE UNE LIGNE
         JSR  CALCADP
         LDY  HTVIR
         LDX  #$00
         STX  LONG
SPCB2    TYA
         PHA
         LDY  LONG
         LDA  (ADP),Y
         STA  SAVA
         PLA
         TAY
         LDA  SAVA
         STA  (ADR),Y
         INY
         INC  LONG
         LDX  LONG
         CPX  #$28
         BCC  SPCB2
         INC  LIGP
SPCLS    INC  LIG
         JSR  AD80
         CLC
         BCC  SPCB1
SPCFIN   LDA  ROM
         RTS
SPCERR   LDX  #$4D
         JMP  ERROR
SPWIN    JSR  CHMAX
         JSR  INPUT
SPWDEB   LDA  LIG
         CMP  VTVIR
         BCC  LIGSUIT
         CLC
         LDA  VTVIR
         ADC  #$17
         CMP  LIG
         BCC  SPWFIN
* AFFICHER UNE LIGNE
         JSR  CALCADP
         LDY  HTVIR
         LDX  #$00
         STX  LONG
SPWB1    LDA  (ADR),Y
         STA  SAVA
         TYA
         PHA
         LDY  LONG
         LDA  SAVA
         STA  (ADP),Y
         PLA
         TAY
         INY
         INC  LONG
         LDX  LONG
         CPX  #$28
         BCC  SPWB1
         INC  LIGP
LIGSUIT  INC  LIG
         JSR  AD80
         CLC
         BCC  SPWDEB
SPWFIN   LDA  ROM
         RTS
SPCHPAG  JSR  ADTX
         JSR  GETBYTC
         CPX  #$01
         BEQ  SPP1
         CPX  #$02
         BEQ  SPP1
         LDX  #$3D
         JMP  ERROR
SPP1     STX  PAGE
         RTS
*** CHARGER MAXV,MAXH
CHMAX    LDA  DIM2
         SEC
         SBC  #$17
         STA  MAXV
         LDA  DIM1+1
         SEC
         SBC  #$27
         STA  MAXH
         RTS
```

## Source 'BIGPGM'

*Assembleur Big Mac, format TEXT*

```
*****************
* PAGE GEANTE
*
* H.ROY-CONTANCIN
* 19/05/86 1.1
****************
VTVIR    EQU  $07
HTVIR    EQU  $08
SAVX     EQU  $09
LOMEM    EQU  $69
ADR      EQU  $18
ADP      EQU  $1A
RAM      EQU  $C083
ROM      EQU  $C082
LIG      EQU  $1C
LIGP     EQU  $1D
LONG     EQU  $1F
INVFLG   EQU  $32
MAXV     EQU  $CE
```

```
MAXH     EQU  $CF
LIGD     EQU  $F9
LIGF     EQU  $FA
COLD     EQU  $FB
COLF     EQU  $FC
SAVY     EQU  $FF
TXTPTR   EQU  $B8
GETBYTC  EQU  $E6F5
DIM1     EQU  $0A
DIM2     EQU  $0C
AD80     EQU  $925A
INPUT    EQU  $9268
SPWDEB   EQU  $92F7
CHMAX    EQU  $934A
ERROR    EQU  $D419
***-CHARGEMENT-ADRESSE
         ORG  $9400
E1       JMP  SPPRINT
E2       JMP  SPCADRE
E3       JMP  SPVIS
E4       JMP  ADTX
INIADR   LDA  #$00
         STA  LIG
         STA  LIGP
         STA  ADR
         LDA  #$D0
         STA  ADR+1
         LDX  SAVX
         LDA  RAM,X
         LDA  RAM,X
         RTS
SPPRINT  LDA  VTVIR
         PHA
         LDA  HTVIR
         PHA
         LDA  DIM2
         STA  MAXV
         LDA  DIM1+1
         STA  MAXH
         BNE  SPPI
         DEC  MAXH
SPPI     JSR  INPUT
         LDA  ROM
         LDY  #$00
         LDA  (LOMEM),Y
         CMP  #$41
         BNE  SPPERR
         INY
         LDA  (LOMEM),Y
         CMP  #$80
         BNE  SPPERR
         INY
         LDA  (LOMEM),Y
         STA  LONG
         INY
         LDA  (LOMEM),Y
         STA  ADP
         INY
         LDA  (LOMEM),Y
         STA  ADP+1
         CLC
         BCC  SPPS
SPPERR   LDX  #$B0
         JMP  ERROR
SPPS     JSR  INIADR
SPPB1    LDA  LIG
         CMP  VTVIR
         BEQ  SPPMOVE
         BCS  SPPFIN
         JSR  AD80
         INC  LIG
         CLC
         BCC  SPPB1
SPPMOVE  LDY  #$00
SPPM1    LDA  (ADP),Y
         ORA  #$80
         CMP  #$C0
         BCS  CFIN
         PHA
         LDA  INVFLG
         CMP  #$7F
         BNE  CN7F
         PLA
         ORA  #$40
CFIN     AND  INVFLG
         PHA
         TYA
         STA  SAVY
         CLC
         ADC  HTVIR
         TAY
         PLA
         STA  (ADR),Y
         LDY  SAVY
         CLC
         BCC  CINY
CN7F     PLA
         CLC
         BCC  CFIN
CINY     INY
         CPY  LONG
         BCC  SPPM1
SPPFIN   PLA
         STA  HTVIR
         PLA
         STA  VTVIR
         LDA  ROM
         RTS
***********
ADTX     LDA  TXTPTR
         CLC
         ADC  #$01
         STA  TXTPTR
         LDA  TXTPTR+1
         ADC  #$00
         STA  TXTPTR+1
         RTS
SPCADRE  JSR  ADTX
         JSR  GETBYTC
         CPX  #$02
         BNE  SPC1
         JSR  OPT1
         JSR  OPT2
         CLC
         BCC  SPKFIN
SPC1     CPX  #$01
         BNE  SPCERR
         JSR  OPT1
SPKFIN   LDA  ROM
         RTS
SPCERR   LDX  #$B0
         JMP  ERROR
*** GRAND CADRE
OPT1     JSR  INIADR
OPT1L1   JSR  TRAIT
OPT1S1   JSR  AD80
         INC  LIG
         LDA  LIG
         CMP  #$5F
         BEQ  OPT1LF
         LDX  DIM1
         CPX  #$00
         BEQ  OPT1S2
         CMP  #$2F
         BEQ  OPT1LF
OPT1S2   LDA  #$20
         LDY  #$00
         STA  (ADR),Y
         LDY  DIM1+1
         DEY
         STA  (ADR),Y
         CLC
         BCC  OPT1S1
OPT1LF   JSR  TRAIT
FINOPT1  LDA  ROM
         RTS
***QUADRILLAGE
OPT2     JSR  GETBYTC
         STX  LIGD
         JSR  GETBYTC
         STX  COLD
         JSR  INIADR
         LDA  #$00
         STA  COLF
         STA  LIGF
         STA  MAXH
OP2B     LDA  LIGD
         CMP  LIGF
         BNE  TRCOL
         JSR  TRAIT
         LDA  #$00
         STA  LIGF
         CLC
         BCC  LIGSUI
TRCOL    LDA  COLD
         CMP  COLF
         BNE  COLSUI
         LDY  MAXH
         LDA  #$20
         STA  (ADR),Y
         LDA  #$00
         STA  COLF
COLSUI   INC  MAXH
         LDA  MAXH
         CMP  DIM1+1
         BEQ  LIGSUI
```

```
          INC   COLF
          CLC
          BCC   TRCOL
LIGSUI    JSR   AD80
          LDA   #$00
          STA   MAXH
          STA   COLF
          INC   LIG
          INC   LIGF
          LDA   LIG
          CMP   DIM2
          BEQ   OPT2FIN
          BNE   OP2B
OPT2FIN   LDA   ROM
          RTS
**
TRAIT     LDA   #$20
          LDY   #$00
TRBL      STA   (ADR),Y
          INY
          CPY   DIM1+1
          BEQ   TRFIN
          BNE   TRBL
TRFIN     RTS
*****VISUALISATION
SPVIS     LDA   DIM1+1
          JSR   CHMAX
          JSR   INPUT
          DEC   MAXH
          DEC   MAXV
          JSR   SPWDEB
SPVISU    INC   $4E
          BNE   BIT1
          INC   $4F
BIT1      BIT   $C000
          BPL   SPVISU
          LDA   $C000
          BIT   $C010
          CMP   #$9B
          BEQ   SPVFIN
          CMP   #$8D
          BEQ   SPVFIN
          CMP   #$88
          BEQ   SPVGAUCHE
          CMP   #$95
          BEQ   SPVDROITE
          CMP   #$8B
          BEQ   SPVHAUT
          CMP   #$8A
          BEQ   SPVBAS
SPWAIT    JSR   INIADR
          JSR   SPWDEB
          CLC
          BCC   SPVISU
SPVGAUCHE LDA HTVIR
          BEQ   SPWAIT
          DEC   HTVIR
          CLC
          BCC   SPWAIT
SPVDROITE LDA HTVIR
          CMP   MAXH
          BEQ   SPWAIT
          INC   HTVIR
          CLC
          BCC   SPWAIT
SPVHAUT   LDA   VTVIR
          BEQ   SPWAIT
          DEC   VTVIR
          CLC
          BCC   SPWAIT
SPVBAS    LDA   VTVIR
          CMP   MAXV
          BEQ   SPWAIT
          INC   VTVIR
          CLC
          BCC   SPWAIT
SPVFIN    STA   $09
          RTS
```

## Source 'BIGSAVE'

*Assembleur Big Mac, format TEXT*

```
*****************
*TRANSFERT-ECRAN
*ENTRE-$5180-ET-CARTE16K
*****************
ADP       EQU   $18
ADR       EQU   $1A
RAM       EQU   $C083
ROM       EQU   $C082
          ORG   $935B
VERMEM    LDA   #$1A
          STA   K1B2+1
          STA   K2B2+1
          LDA   #$18
          STA   K1B2+3
          STA   K2B2+3
          JMP   TRANS
VER16K    LDA   #$18
          STA   K1B2+1
          STA   K2B2+1
          LDA   #$1A
          STA   K1B2+3
          STA   K2B2+3
TRANS     LDA   #$51
          STA   ADP+1
          LDA   #$80
          STA   ADP
          LDA   #$00
          STA   ADR
          LDA   #$D0
          STA   ADR+1
BANK1     LDA   RAM
          LDA   RAM
          LDX   #$00
K1B1      LDY   #$00
K1B2      LDA   (ADR),Y
          STA   (ADP),Y
          INY
          BNE   K1B2
          INC   ADR+1
          INC   ADP+1
          INX
          CPX   #$30
          BNE   K1B1
BANK2     LDX   #$08
          LDA   RAM,X
          LDA   RAM,X
          LDA   #$00
          STA   ADR
          LDA   #$D0
          STA   ADR+1
          LDX   #$00
K2B1      LDY   #$00
K2B2      LDA   (ADR),Y
          STA   (ADP),Y
          INY
          BNE   K2B2
          INC   ADR+1
          INC   ADP+1
          INX
          CPX   #$10
          BNE   K2B1
FIN       LDA   ROM
          RTS
```

## Schémateur... suite

Le programme de dessins de graphes et fonctions du numéro 31 de Pom's mérite un petit *patch* pour permettre la lecture des fichiers sur des disquettes ne contenant pas la routine de chargement rapide :

- Dans le programme Schémateur,

Ajouter :

```
2365 PRINT D$"BLOADCHARGE.
   OBJ,A$9000"
```

Modifier :

```
2100 CALL 36864
2390 CALL 36864
```

- Dans le programme Gravure,

Ajouter :

```
145 PRINT D$"BLOADCHARGE.
   OBJ,A$9000"
```

Modifier :

```
160 CALL 36864 : &"GRAV.C"
   ,AG
260 HGR : POKE 49234,0 : C
   ALL 36864 : &IM$,8192
```

# Récapitulation BIGPTR

*Après avoir saisi cette table sous moniteur, vous la sauvegarderez par BSAVE BIGPTR,A$300,L$30*

```
0300:00 04 80 04 00 05 80 05
0308:00 06 80 06 00 07 80 07
0310:28 04 A8 04 28 05 A8 05
0318:28 06 A8 06 28 07 A8 07
0320:50 04 D0 04 50 05 D0 05
0328:50 06 D0 06 50 07 D0 07
```

# Récapitulation 'ECR-VIRT'

Cette récapitulation regroupe les objets issus des trois sources et la table BigPtr

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par BSAVE ECR-VIRT,A$9180,L$480*

```
9180:A9 4C 8D F5 03 A9 90 8D
9188:F6 03 A9 91 8D F7 03 60
9190:A8 A5 0A F0 0F A9 00 85
9198:0B A9 01 85 0A A9 30 85
91A0:0C 18 90 08 A9 80 85 0B
91A8:A9 60 85 0C A5 06 C9 02
91B0:F0 07 A9 00 85 09 18 90
91B8:04 A9 08 85 09 98 C9 48
91C0:F0 1D C9 43 F0 1C C9 57
91C8:F0 1B C9 50 F0 1A C9 4B
91D0:F0 19 C9 54 F0 18 C9 56
91D8:F0 17 A2 10 4C 19 04 4C
91E0:F4 91 4C A9 92 4C F1 92
91E8:4C 00 94 4C 03 94 4C 34
91F0:93 4C 06 94 20 09 94 20
91F8:F5 E6 86 F9 8A C5 0C B0
9200:2F 20 F5 E6 E4 0C B0 28
9208:E8 86 FA 8A C5 F9 90 20
9210:20 F5 E6 86 FB A5 0B F0
9218:04 E4 0B B0 13 20 F5 E6
9220:86 FC A5 0B F0 04 E4 0B
9228:B0 06 E4 FB 90 02 B0 05
9230:A2 4D 4C 19 D4 20 0C 94
9238:A2 00 E4 F9 90 08 E4 FA
9240:90 0B AD 82 C0 60 E8 20
9248:5A 92 18 90 ED A4 FB A9
9250:A0 91 18 C4 FC C8 90 F9
9258:B0 EC 18 A5 18 65 0B 85
9260:18 A5 0A 65 19 85 19 60
9268:20 09 94 20 F5 E6 86 07
9270:E4 CE B0 78 20 F5 E6 86
9278:08 E4 CF B0 6F A9 00 85
9280:1D 85 1C 85 18 A9 D0 85
9288:19 A6 09 BD 83 C0 BD 83
9290:C0 A9 D0 85 FD A9 93 85
9298:FE 60 A5 1D 18 0A A8 B1
92A0:FD 85 1A C8 B1 FD 85 1B
92A8:60 20 4A 93 20 68 92 A5
92B0:1C C5 07 90 2B 18 A5 07
92B8:69 17 C5 1C 90 2A 20 9A
92C0:92 A4 08 A2 00 86 1F 98
92C8:48 A4 1F B1 1A 85 FF 68
92D0:A8 A5 FF 91 18 C8 E6 1F
92D8:A6 1F E0 28 90 E9 E6 1D
92E0:E6 1C 20 5A 92 18 90 C7
92E8:AD 82 C0 60 A2 4D 4C 19
92F0:D4 20 4A 93 20 68 92 A5
92F8:1C C5 07 90 2B 18 A5 07
9300:69 17 C5 1C 90 2A 20 9A
9308:92 A4 08 A2 00 86 1F B1
9310:18 85 FF 98 48 A4 1F A5
9318:FF 91 1A 68 A8 C8 E6 1F
9320:A6 1F E0 28 90 E9 E6 1D
9328:E6 1C 20 5A 92 18 90 C7
9330:AD 82 C0 60 20 09 94 20
9338:F5 E6 E0 01 F0 09 E0 02
9340:F0 05 A2 3D 4C 19 D4 86
9348:06 60 A5 0C 38 E9 17 85
9350:CE A5 0B 38 E9 27 85 CF
9358:60 FB 00 A9 1A 8D 99 93
9360:8D BD 93 A9 18 8D 9B 93
9368:8D BF 93 4C 7E 93 A9 18
9370:8D 99 93 8D BD 93 A9 1A
9378:8D 9B 93 8D BF 93 A9 51
9380:85 19 A9 80 85 18 A9 00
9388:85 1A A9 D0 85 1B AD 83
9390:C0 AD 83 C0 A2 00 A0 00
9398:B1 1A 91 18 C8 D0 F9 E6
93A0:1B E6 19 E8 E0 30 D0 EE
93A8:A2 08 BD 83 C0 BD 83 C0
93B0:A9 00 85 1A A9 D0 85 1B
93B8:A2 00 A0 00 B1 1A 91 18
93C0:C8 D0 F9 E6 1B E6 19 E8
93C8:E0 10 D0 EE AD 82 C0 60
93D0:00 04 80 04 00 05 80 05
93D8:00 06 80 06 00 07 80 07
93E0:28 04 A8 04 28 05 A8 05
93E8:28 06 A8 06 28 07 A8 07
93F0:50 04 D0 04 50 05 D0 05
93F8:50 06 D0 06 50 07 D0 07
9400:4C 21 94 4C B9 94 4C 75
9408:95 4C AB 94 A9 00 85 1C
9410:85 1D 85 18 A9 D0 85 19
9418:A6 09 BD 83 C0 BD 83 C0
9420:60 A5 07 48 A5 08 48 A5
9428:0C 85 CE A5 0B 85 CF D0
9430:02 C6 CF 20 68 92 AD 82
9438:C0 A0 00 B1 69 C9 41 D0
9440:19 C8 B1 69 C9 80 D0 12
9448:C8 B1 69 85 1F C8 B1 69
9450:85 1A C8 B1 69 85 1B 18
9458:90 05 A2 B0 4C 19 D4 20
9460:0C 94 A5 1C C5 07 F0 0A
9468:B0 37 20 5A 92 E6 1C 18
9470:90 F0 A0 00 B1 1A 09 80
9478:C9 C0 B0 0A 48 A5 32 C9
9480:7F D0 15 68 09 40 25 32
9488:48 98 85 FF 18 65 08 A8
9490:68 91 18 A4 FF 18 90 04
9498:68 18 90 EA C8 C4 1F 90
94A0:D3 68 85 08 68 85 07 AD
94A8:82 C0 60 A5 B8 18 69 01
94B0:85 B8 A5 B9 69 00 85 B9
94B8:60 20 AB 94 20 F5 E6 E0
94C0:02 D0 09 20 DC 94 20 0C
94C8:95 18 90 07 E0 D1 D0 07
94D0:20 DC 94 AD 82 C0 60 A2
94D8:B0 4C 19 D4 20 0C 94 20
94E0:67 95 20 5A 92 E6 1C A5
94E8:1C C9 5F F0 18 A6 0A E0
94F0:00 F0 04 C9 2F F0 0E A9
94F8:20 A0 00 91 18 A4 0B 88
9500:91 18 18 90 DD 20 67 95
9508:AD 82 C0 60 20 F5 E6 86
9510:F9 20 F5 E6 86 FB 20 0C
9518:94 A9 00 85 FC 85 FA 85
9520:CF A5 F9 C5 FA D0 0A 20
9528:67 95 A9 00 85 FA 18 90
9530:1D A5 FB C5 FC D0 0A A4
9538:CF A9 20 91 18 A9 00 85
9540:FC E6 CF A5 CF C5 0B F0
9548:05 E6 FC 18 90 E3 20 5A
9550:92 A9 00 85 CF 85 FC E6
9558:1C E6 FA A5 1C C5 0C F0
9560:02 D0 BE AD 82 C0 60 A9
9568:20 A0 00 91 18 C8 C4 0B
9570:F0 02 D0 F7 60 A5 0B 20
9578:4A 93 20 68 92 C6 CF C6
9580:CE 20 F7 92 E6 4E D0 02
9588:E6 4F 2C 00 C0 10 F5 AD
9590:00 C0 2C 10 C0 C9 9B F0
9598:45 C9 8D F0 41 C9 88 F0
95A0:15 C9 95 F0 1A C9 8B F0
95A8:21 C9 8A F0 26 20 0C 94
95B0:20 F7 92 18 90 CE A5 08
95B8:F0 F3 C6 08 18 90 EE A5
95C0:08 C5 CF F0 E8 E6 08 18
95C8:90 E3 A5 07 F0 0F C6 07
95D0:18 90 DA A5 07 C5 CE F0
95D8:D4 E6 07 18 90 CF 85 09
95E0:60 90 CF 85 09 60 00 00
95E8:BB BB 00 00 BB FB 00 00
95F0:BB BB 00 00 BB FB 00 00
95F8:BB BB 00 00 BB FB B0 04
```

# Un détecteur de sonnerie

DOS 3.3
ProDOS

*Paul Courbis*

Le but de ce montage est de permettre à votre Apple favori de détecter la sonnerie du téléphone afin de provoquer l'exécution d'un logiciel (serveur, répondeur télématique, compteur d'appels, envoi ou réception de fichiers etc.). La présence de la sonnerie se traduit par la fermeture du bouton joystick n° 0 c'est-à-dire . La sonnerie détectée, le programme d'application devra se charger de décrocher la ligne, connecter le modem ou autre.

## Principe

Le courant de sonnerie est un signal alternatif à 100 Hz. Le condensateur 2,2 µF 'filtre' ce signal qui est réduit à une tension raisonnable par la résistance de 390 Ω. Le pont de diodes et le condensateur redresse le signal qui devient continu et active le relais. Ce dernier se comporte comme un bouton de joystick et est connecté avec une résistance comme indiqué dans les Manuels de Référence Apple. À noter que l'Apple est à l'abri des soucis puisqu'il est isolé du réseau téléphonique par le relais.

## Le programme

Dans le programme, il suffit de tester l'octet (–16287) : une valeur égale ou supérieur à 128 indique que la sonnerie est en cours. En assembleur, le test du bit 7 donne l'indication : à 1 le téléphone sonne, à 0, il ne sonne pas. À noter qu'on peut simuler la sonnerie en pressant la touche .
On peut s'inspirer du programme DETECT qui attend un certain temps avant de signaler la sonnerie : celà évite la détection des tintements parasites tels celui dû au raccroché du combiné.

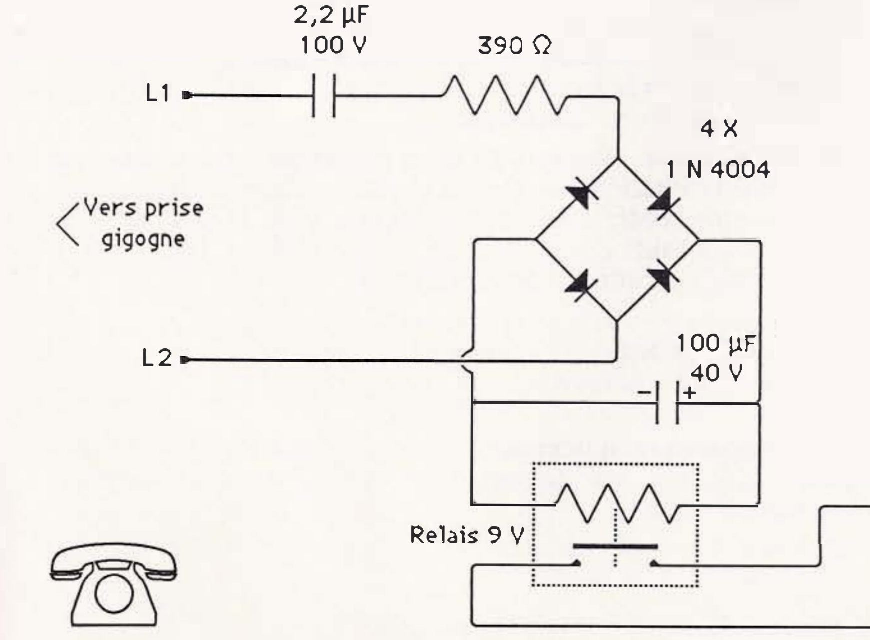
## En pratique

Le petit montage s'effectuera sans difficulté sur une plaque d'essai. L1 et L2 sont à relier à la prise gigogne standard intercalée entre la prise murale du téléphone et l'appareil téléphonique. Ces fils sont à connecter aux bornes 1 et 3. Il ne faut pas oublier de relier les bornes des côtés mâle et femelle pour que l'appareil téléphonique fonctionne toujours. Côté Joystick, la numérotation de la prise correspond à celle de la DB 9 à l'arrière de l'ordinateur.



## Programme DETECT

```
10 A$ = "Le téléphone ne s
     onne pas
20 B$ = "Le téléphone sonn
     e      "
30 HOME
40 PRINT : PRINT : PRINT
     : PRINT "Etat de la li
     gne :"
50 FOR W = 1 TO 10: VTAB
     10: HTAB 1
60 VTAB 10: HTAB 1
70 IF PEEK ( - 16287) >
     127 THEN NEXT : PRINT
     B$: GOTO 50
80 PRINT A$:W = 1: GOTO 6
     0
```



# *Essai* Unimate

L'objet d'**Unimate**, programme de Cirtech, est d'utiliser les programmes DOS 3.3, Pascal ou CP/M sur les Unidisks 3'5. Il permet de faire fonctionner sur un support plus plaisant et spacieux des applications allergiques à ProDOS ou difficilement transposables : voici un programme qui mériterait d'accompagner chaque lecteur 800Ko...

Unimate est livré sur une disquette lisible en DOS 3.3, en Pascal et en CP/M. Il est compatible avec tous les Apple //. Depuis chacun des systèmes d'exploitation, les fichiers concernés sont copiables directement.

Un démarrage sur la disquette Unimate donne accès à un mode d'emploi imprimable en anglais. Il est succinct car à part l'installation des *drivers*, l'utilisation est totalement transparente.

## DOS 3.3

La méthode retenue est l'installation d'une modification au DOS. Il convient de démarrer en DOS 3.3, puis d'exécuter le programme livré : le DOS en mémoire est alors modifié et il suffit d'initialiser des disquettes 3'5 (ou 5'25) qui comporteront le nouveau DOS ; elles sont *bootables*.

Chaque Unidisk 3'5 est reconnu en DOS 3.3 comme deux disques de 400Ko. Deux Unidisks en Slot 5 seraient reconnus comme étant :

S5, D1
S5,D2
S5,D3
S5, D4

Seule précaution, pour initialiser le disque du deuxième lecteur, faire `INIT HELLO, D3` et non pas `D2`, sinon...
Les deux faces s'initialisent alors simultanément et le `HELLO` est installé en D3.

Une limitation : le numéro de volume n'est plus vérifié sur l'Unidisk.

Un FID adapté (nommé UNIFID) est également livré pour permettre les copies sur disques 3'5.

## Pascal 1.1, 1.2

Les 'drivers' livrés permettent à Pascal 1.1 de reconnaître deux Unidisks comme étant les volumes 9 et 10 de 800Ko chacun. Pour la version 1.2, ce seront les volumes 19 et 20.

Un nouveau FORMATTER livré autorise le formatage des disques 5'25 et 3'5.

## CP/M 2.23, 2.20B

Le driver nécessaire est installé par l'exécution d'un programme lors du démarrage.

Pour CP/M 2.23, les Unidisk seront les volumes `C:` et `D:` ; `E:` et `F:` pour la version 2.20B. Chaque volume Unidisk comporte 788Ko.

Un programme de formatage est également livré.

## Speedisk

Grâce à Unimate, il est possible d'utiliser la carte Ram Speedisk sous DOS 3.3. La carte SP1000 de 1 méga-octet se trouve alors reconnue comme les Unidisks : on dispose de deux lecteurs virtuels de 400 Ko sous DOS 3.3.

Place faite au DOS et au catalogue (qui accueille 216 fichiers), il reste disponible 2 fois 376 832 octets pour les fichiers sur la carte. Un "plus" pour les applications allergiques à ProDOS.

*Unimate est distribué par Alpha Systèmes, 29, bld Gambetta 38000 Grenoble.*



# Démarrer sur l'Unidisk 800 Ko ?

Effectivement, l'Apple //e ne considère pas que la carte contrôleur de l'Unidisk 800Ko est susceptible de le faire démarrer, même si cette carte est dans le port 6 et les lecteurs 140Ko en port 5. Pourtant, neuf fois sur dix, on souhaite utiliser le lecteur 3'5 ce qui conduit à un CTRL-RESET, suivi d'un PR£5 au clavier : pas élégant.

On peut laisser sur le lecteur 140Ko un programme Basic du style PRINT CHR$(4)"PR£5" mais c'est lent. La solution la plus rapide semble de mettre l'ordre PR£5 sur le secteur 0 de la piste 0 de la disquette 5'1/4 : à l'aide d'un éditeur de secteur (Bag of Tricks, Copy ][+, Mobby Disk...), mettre les octets suivants :

```
01                         convention
A6 2B       LDX $2B        arrête le lecteur 5 1/4
BD 88 C0    LDA $C088,X
4C 00 C5    JMP $C500      = PR£5
```

Le dernier octet est à remplacer par C4 pour un démarrage sur le port 4 par exemple. Cette disquette de démarrage pourra alors rester à demeure dans le lecteur 140 Ko.



# Commande XCAT :
# TOUT le catalogue ProDOS

Sylvie Gallet

XCAT, commande externe ProDOS a pour objet de lister tous les fichiers d'un volume, qu'ils se trouvent au niveau 0 ou dans un énième sous-dossier. Il est écrit en 6502 pour fonctionner sur tous Apple // sous ProDOS versions 1.1 et suivantes ; pour les versions antérieures de ProDOS, il faudra modifier le relogeur pour qu'il recherche lui-même la place en mémoire.

Le source, abondamment commenté, contient toutes les explications nécessaires à son utilisation mais il reste quelques précisions à apporter.

## L'assembleur ORCA/M

- Le source occupe environ de 50 % de la mémoire disponible (environ 36 Ko) il peut donc être saisi en une seule fois, confort non négligeable ;
- si un commentaire est placé après une instruction, le point virgule n'est pas obligatoire ;
- `anop` permet de ne pas écrire d'instruction après un label (anop ne génère pas de code: c'est l'équivalent de ':' pour LISA) ;
- `keep xcat` : le code généré est sauvegardé automatiquement sous le nom 'xcat' (équivalent du DSK de ProCODE) ;
- `£<` et `£>` signifient respectivement partie basse et partie haute ;
- `ds n` : réserve n octets en mémoire et les initialise à zéro ;

\- `dc i1'n1,n2,..,np'` : définit les entiers n1, n2, ..., np chacun sur un octet ;

\- `dc i2'label'` : l'entier label est stocké sur deux octets (bas, haut). Cette instruction permet de stocker l'adresse label ;

\- `dc c'chaîne'` : définit la chaîne de caractères 'chaîne' (bits 7 à 0) ;

- les instructions implicites `ASL`, `LSR`, `ROR`, `ROL` doivent être suivies de `A`.

## Le fonctionnement de XCAT

- On commence par vérifier si la commande est bien XCAT : si ce n'est pas le cas, on en fait cadeau aux éventuelles autres commandes externes ;
- si c'est pour nous : on demande à ProDOS d'analyser les paramètres qui peuvent être soit un nom de volume, soit un numéro de slot et/ou drive ;
- si un nom de volume est demandé, on utilise ON LINE pour savoir dans quelle unité il se trouve. S'il existe, on met le numéro de l'unité dans la table des paramètres de READ BLOCK ;

\- si S ou D est utilisé, on calcule le numéro de l'unité et on termine comme ci-dessus ;

- on lit le bloc 2 de l'unité choisie, on vérifie qu'il s'agit bien d'un disque ProDOS ; dans l'affirmative, l'analyse peut commencer ;
- celle-ci se fait très simplement en lisant l'un après l'autre les blocs de ce catalogue et en analysant les 13 entrées de chaque bloc. Un catalogue ou sous-catalogue est terminé quand le chaînage avant est égal à 00 00.

Au cours de l'analyse d'un catalogue, si on trouve le nom d'un sous-catalogue, on sauve 3 octets sur la pile : le numéro du bloc en cours et le numéro de l'enregistrement, on procède ensuite à l'analyse du sous-catalogue : quand celui-ci est terminé, on dépile les 3 octets sauvés et on reprend le catalogue

### Exemple d'exécution

```
$XCAT /SPDK/

CATALOGUE IMPRIME LE :   9-SEP-87

NOM                          TYPE  BLOCS  MODIFIE
-------------------------------------------------

SPDK
----
  MERLIN
  ------
    LIB
    ---
      ESSAI
      -----
      SENDMSG.S              TXT      1  27-AUG-87
      OUTPUT.S               TXT      3  27-AUG-87
    UTIL
    ----
      XREF                   BIN      5  27-AUG-87
      MON.65C02              BIN      1  27-AUG-87
    SOURCE
    ------
      PI.START.S             TXT      8  27-AUG-87
      PI.MAIN.S              TXT      7  27-AUG-87
      PI.LOOK.S              TXT      3  27-AUG-87
    PI
    --
      START                  SF8      9  27-AUG-87
      MAIN                   SF8      1  27-AUG-87
      LOOK                   SF8      1  27-AUG-87
      DIV                    SF8      1  27-AUG-87
    MERLIN.SYSTEM            SYS     37  27-AUG-87
    PARMS                    BIN      1  27-AUG-87
    PA.S                     TXT      1  27-AUG-87
    PA                       BIN      1  27-AUG-87
  CLV
  ---
    PROC                     TXT      1  27-AUG-87
    CP13                     TXT      1  27-AUG_87
    GH20                     TXT      1  27-AUG-87
    MAB10                    TXT      1  27-AUG-87
    C1708                    TXT      3  27-AUG-87
  PRODOS                     SYS     32  31-AUG-87
  REBOOT                     SYS      4  27-AUG-87
  FILER                      SYS     52  31-AUG-87
  KRUPTOS                    BIN      3  31-AUG-87
  STARTUP                    BAS      1   9-SEP-87
  INTERPOMS.v2               BIN     17   7-SEP-87
  COPIERAM                   TXT     11  31-AUG-87
  T.POMS                     BIN      9  31-AUG-87
  XCAT                       BIN      5   9-SEP-87
  BASIC.SYSTEM               SYS     21  31-AUG-87
  CLV.POMS                   BIN     10   9-SEP-87
......

BL LIBRES :   615  OCCUPES :  1433   TOTAL :  2048

$
```

précédent là où on l'avait laissé.

L'utilisation de la pile permet une sauvegarde très simple des valeurs indispensables à la reprise d'un catalogue sans perte de place dans le programme et avec un contrôle très simple de la profondeur : le pointeur de pile étant sauvegardé au début, il suffit à la fin d'un sous-catalogue de comparer le pointeur actuel et le premier pour savoir si tout a été dépilé et si on est dans le catalogue principal.

### Remarque sur la page zéro

La plupart des adresses libres en page zéro sont utilisées. Si on souhaite préserver le contenu de ces adresses, il est possible de sauver la page zéro dans le buffer de ProDOS (la moitié supérieure de ce buffer n'est pas utilisée par READ BLOCK).

## Structure des catalogues et de la bitmap

Le catalogue principal comporte 4 blocs : n° 2, 3, 4 et 5. Chaque bloc est chaîné au précédent et au suivant.

Les sous-catalogues comportent au minimum 1 bloc auquel peuvent être chaînés d'autres blocs si le nombre de fichiers est supérieur à 13.

Chaque bloc comporte :

octets $0 et $1 : 2 octets indiquant le n° du bloc qui le précède (c'est le chaînage arrière): contiennent 00 00 si ce bloc est le premier ;

octets $2 et $3 : numéro du bloc suivant (chaînage avant) : 00 00 si ce bloc est le dernier ;

la suite du bloc est partagée en 13 entrées de fichier de $27 octets chacune ;

il reste 1 octet inutilisé.

La première entrée du bloc 2 ainsi que la première entrée de chaque premier bloc de sous-catalogue contiennent des informations relatives à la disquette ou au sous-catalogue et sont légèrement différentes des entrées de fichiers.

### Bloc 2 1ère entrée

(octets $4 à $2A). Elle contient l'en-tête du catalogue principal.

XCAT utilise les octets suivants :

**$4**
- type de fichier (4 1ers bits) :
  $0 fichier effacé ou inexistant ou effacé,
  $D fichier sous-catalogue,
  $E en-tête sous-catalogue,
  $F en-tête catalogue.
- longueur du nom de la disquette (4 derniers bits d'où la limitation à 15 caractères) ;

**$5 à $13.**
nom de la disquette ;

**$27 et $28**
numéro de bloc de la bitmap ;

**$29 et $2A**
nombre maximum de blocs du support.

### 1er bloc de sous-catalogue, 1ère entrée

On utilise seulement les octets $4 à $13 (comme ci-dessus).

### Entrées de fichiers

(Les numéros d'octets sont donnés par rapport au début de l'enregistrement). Les octets utilisés sont les suivants :

**$0**
type de fichier, longueur du nom (voir ci-dessus) ;

**$1 à $F**
le nom ;

**$10**
type de fichier (BAS, BIN, TXT, ...) ;

**$11 et $12**
numéro du 1er bloc du fichier ;

**$13 et $14**
nombre de blocs occupés par le fichier ;

**$21 et $22**
date de dernière modification du fichier (à lire à l'envers $22, $21).

### La bitmap

Elle occupe généralement le bloc 6. Chaque octet renseigne sur l'occupation de 8 blocs : un bit est à 1 si le bloc est libre et à 0 si le bloc est occupé.

Sur une disquette 140 Ko il y a donc 35 octets utilisés (280/8) et 200 octets pour une disquette 800 Ko (1600/8).



## *Mode d'emploi*

*Si vous n'avez pas la disquette d'accompagnement Pom's, il vous faut saisir et sauvegarder le code XCAT.*
*La commande externe est disponible après avoir tapé en mode direct:*

```
- XCAT
```

*ou, par programme :*

```
PRINT CHR$(4)"-XCAT".
```

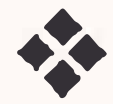
*Pour obtenir le catalogue étendu, taper simplement :*

```
XCAT /nom de volume/ OU
XCAT ,Ss ,Dd
```

*s & d étant le numéro de port et de lecteur.*

*Le listing peut être suspendu et repris à l'aide de la barre d'espace ; il est stoppé par CTRL-C.*





## Bibliographie


Organisation d'une disquette ProDOS :

- Guide ProDOS P. Beaufils W. Luther, Éditions Sybex
- Beneath Apple ProDOS, Quality Software

Relogeur, commande externe :

- numéros antérieurs de Pom's.




## Source XCATS
## Assembleur ORCA/M

```
; ------------------------------------------------------------------
;    commande externe XCAT permettant de lister le catalogue
;         et les sous-catalogues d'une disquette ProDOS
;
;         © Sylvie GALLET pour POM'S  Juillet 1987
; ------------------------------------------------------------------

;                    Assembleur ORCA/M

           list on
           err on
           65816 off
           65C02 off
           absaddr on
           keep /Z/XCAT

           org $2000

main       START

; EQUATES:
; -------

; adresses en page zéro:
; ---------------------
himem     gequ $73
length    gequ $2F         longueur de l'instruction
pcl       gequ $3A         adresse de la ligne désassemblée
a1        gequ $3C         adresse de départ pour move
a2        gequ $36         adresse de fin     "     "
a4        gequ $42         adresse d'arrivée  "     "

nbloc     gequ $6          nombre sur 2 octets (bas,haut)
temp      gequ $8
longueur  gequ $1A         longueur de nom de volume
debsscat  gequ $1B         indicateur de soulignement
compteur  gequ $1C         compteur d'enregistrements dans cat.
increm    gequ $1D         décalage de début de ligne
pile      gequ $1E         sauvegarde du pointeur de pile
htab      gequ $1F         position horizontale du curseur
carte80c  gequ $E3         flag : carte 80 col active (pour éviter
;                          de lui envoyer un form-feed
bitmap    gequ $EB         numéro de bloc de la bitmap (bas, haut)
blocmax   gequ $ED         nbre max. de blocs du support (bas, haut)
lignes    gequ $EF         nombre de lignes affichées
aa        gequ $F9         année
mm        gequ $FA         mois
jj        gequ $FB         jour
adr       gequ $FC         adresse temporaire
ptr       gequ $FE            "         "

; sous-programmes moniteur et adresses système:
; ---------------------------------------------

linprint  gequ $ED24       affiche en décimal le contenu de A,X
insdsp2   gequ $F88C       désassemble la ligne (pcl)
move      gequ $FE2C       comme son nom l'indique
prbyte    gequ $FDDA       affiche (A) en 2 chiffres hexa
kbd       gequ $C000       lecture directe clavier
kbstrobe  gequ $C010       initialisation clavier

; ProDOS et BASIC.SYSTEM:
; ----------------------

typnum    gequ $B989       codes des types de fichier ayant une
;                          abréviation
typtable  gequ $B997       table des abréviations
mois      gequ $B9C1       table des noms des mois
mli       gequ $BF00       point d'entrée de ProDOS
extrncmd  gequ $BE06       saut vers commande externe
errout    gequ $BE09       affiche erreur et fin
badcall   gequ $BE8B       convertit code erreur MLI en BASIC.S
getbufr   gequ $BEF5       réserve (A) pages sous BASIC SYSTEM
xtrnaddr  gequ $BE50       adresse commande externe
xlen      gequ $BE52       longueur de la commande
xcnum     gequ $BE53       n° de la commande
pbits     gequ $BE54       paramètres autorisés
fbits     gequ $BE56       paramètres effectivement trouvés
vpath1    gequ $BE6C       vecteur vers la commande entrée
syserr    gequ $BF0F
xreturn   gequ $BE9E       un RTS qui gagne à etre connu
vslot     gequ $BE61       n° slot trouvé
vdriv     gequ $BE62       "  drive   "
opencnt   gequ $BE4D       nombre fichiers ouverts
vpath2    gequ $BE6E       2-ème nom de fichier
devcnt    gequ $BF31       nombre de lecteurs
output    gequ $BE31       octet haut adresse du périph de sortie
datesys   gequ $BF90       date courante

long      gequ fin-debut+$100 longueur du programme à reloger

; ------------------------------------------------------------------
;                          relogeur
; ------------------------------------------------------------------

;  Relogeur très semblable à celui de POM'S n° 26 et
; légèrement modifié pour reloger les appels au MLI

;  Il se charge d'installer la commande externe entre ProDOS
; et ses buffers.

;  Getbufr n'étant disponible qu'avec les versions ProDOS 1.1
; et suivantes, une petite modification sera nécessaire pour
; les versions antérieures.

; initialisation
; --------------

init       lda mli
           cmp £$4C               est-ce un JMP ?
           beq init1              oui
           ldy nonactif           non -> affiche message et fin
nabcl      lda nonactif,y
           jsr outdo
           dey
           bne nabcl
           rts

init1      lda opencnt            y a-t-il des fichiers ouverts ?
           beq init2              non
           lda £$15               oui -> impossible placer la commande
;                                 message FILE(S) STILL OPEN
           jmp errout

init2      lda extrncmd+2         sauve adresse de la première
           sta precmd+2           commande externe
           lda extrncmd+1
           sta precmd+1
           lda £>long             demande d'un buffer de (A) pages
           jsr getbufr
           cmp £$0C               erreur ?
           bne gothen             non
           jmp errout             oui -> affiche 'NO BUFFERS AVAILABLE'
;                                        et fin

gothen     sta adresse            n° de la 1-ère page accordée par
           sta a4+1               ProDOS
           sta extrncmd+2
           lda £>debut
           sta a1+1
           lda £<fin-1
           sta a2
           lda £>fin-1
           sta a2+1
           ldy £0
           sty a4                 les adresses de départ et d'arrivée
           sty a1                 sont alignées -> poids faible = 0
           sty extrncmd+1
           jsr move               déplace le programme
           jsr reloge
           rts

reloge     lda adrease            adresse de relogement
           sta pcl+1                  "
           lda £0                     "
           sta pcl                    "
rdecode    ldx £0                 nécessaire pour insdsp2
           lda (pcl,x)
           beq rfin               si 00 séparant programme et données
           jsr insdsp2
           ldy length             Y = longueur instruction - 1
           cpy £2                 instruction sur 3 octets ?
           bne rsuivant           non -> pas de changement
           lda (pcl),y            oui
           cmp £>mli              est-ce un appel au MLI ?
           bne nomli              non
           dey                    peut-etre...
           lda (pcl),y
           cmp £<mli
           bne nomli              non
           dey
           lda (pcl),y
           cmp £$20               est-ce un jsr ?
           bne nomli              non
           lda pcl                oui -> avance pcl de 3 octets et
           clc                    traite les 3 octets suivant jsr $BF00
           adc £3                 comme une instruction à reloger
;                                 ( l'adresse du buffer n'a pas besoin
;                                 d'etre relogée )
           sta pcl
           bcc nomli
           inc pcl+1
nomli      lda pcl                passage de paramètres pour calcul
           sta adr
           lda pcl+1
           sta adr+1
           ldy length
           jsr calcul
rsuivant   lda length             on passe à l'instruction suivante:
           sec                    avec pcl = pcl + length
           adc pcl
           sta pcl
           lda pcl+1
           adc £0
```

][+
//e
//e+
//c
][gs

ProDOS

```
         sta pcl+1
         sec                  distance (pcl) - adresse stockée dans
         sbc adresse          A (MSB) et X (LSB)
         ldx pcl
         jsr test             distance >= longueur du programme ?
         bcs rfin             oui -> on a fini
         jmp rdecode          non
rfin     rts

test     cmp £>fin-debut      A < nbre de pages du programme ?
         bcc oui              oui
         bne non              non et A > nbre de pages -> en dehors
;                                  du programme
         cpx £<fin-debut      on est dans la dernière page:
;                               X < poids faible de fin ?
         bcs non              non -> en dehors du programme
oui      clc                  dans les limites
         rts
non      sec
         rts

calcul   dey
         lda (adr),y          partie basse de l'opérande
         tax
         iny
         lda (adr),y          partie haute de l'opérande
         sec
         sbc £>debut          écart (en pages) entre adresse et début
         bcc nonc             écart < 0 -> opérande en dehors des
;                                  adresses à modifier
         jsr test             écart >= 0 -> écart > longueur du
;                                   programme ?
         bcs nonc                   non
         adc adresse                oui -> on reloge
         sta (adr),y
         clc
         rts
nonc     sec
         rts

adresse  ds 1                 adresse de début du programme relogé

         ds $FF-(adresse-init)     debut sera égale à $2100

; ------------------------------------------------------------
;                      A propos de XCAT
; ------------------------------------------------------------

;        fonction de XCAT:
;        -----------------

;  catalogue étendu de tout support ProDOS, disponible en mode
; direct ou différé.
;  - les noms de tous les fichiers sont affichés avec une indentation
; mettant en évidence les sous-catalogues.
;  - Il n'y a pas de limitation du nombre de niveaux d'emboitement
; des sous-catalogues (sauf le bon sens), l'indentation est
; seulement limitée à 24 caractères.
;  - Les noms de catalogues sont soulign
;  - Les autres fichiers sont suivis du type, du nombre de blocs
;    et de la date de dernière modification.
;  - le nombre de blocs libres, occupés, total est affiché meme
;    si xcat est interrompu avant la fin.
;  - La date courante est affichée en tete du catalogue.

;        installation:
;        ------------

;  'brun xcat' ou '-xcat'

;        utilisation:
;        -----------

;  syntaxe: en mode direct: xcat /nomvolume/ , Ss , Dd
;           en mode différé: print chr$(4)"xcat" suivi éventuellement
;                            de paramètres (constantes ou variables
;                            chaine).

;  les paramètres sont optionnels : si on les omet, le lecteur
; par défaut sera utilisé.

;  la présence de paramètres meme S et D ne modifie pas le prefixe
; par défaut.

;  Arret temporaire et reprise du défilement avec une touche
; quelconque.

;  arret définitif par CTRL-C: dans ce cas, on affiche '.....'
; pour signaler qu'il reste peut-etre des fichiers.

;  la sortie se fait sur le périphérique en service ( écran 40
; ou 80 colonnes ou imprimante ), elle est évidemment plus
; claire en 80 colonnes. A l'imprimante, un saut de page est
; envoyé toutes les 60 lignes environ.

;        messages d'erreur possibles :
;        ----------------------------

; FILE(S) STILL OPEN : pour installer XCAT, tous les fichiers
;                      doivent etre fermés.
; I/O ERROR          : erreur de lecture disque ( bloc
;                      endommagé, lecteur ouvert, disque non
;                      ProDOS...).
; SYNTAX ERROR       : erreurs diverses dont un nom de volume
;                      incorrect ( trop ou pas assez de / ,
;                      nom de la forme / ou // , nom contenant
;                      des caractères non autorisés ).
; PATH NOT FOUND     : le volume n'est pas en ligne.
; RANGE ERROR        : valeur en dehors des limites autorisées.
; INVALID PARAMETER  : si on utilise d'autres paramètres que
;                      slot ou drive.
; PROGRAM TOO LARGE  : peu probable.
; NO BUFFER AVAILABLE:  "     "
; PRODOS NON ACTIF   :  "     "

; ------------------------------------------------------------
;                  programme à reloger
; ------------------------------------------------------------

debut     cld                  nécessaire... pour faire reler Yvan Koenig !!
          lda vpathl           vpathl pointe sur la commande
          sta ptr              entrée
          lda vpathl+1
          sta ptr+1
          ldy £1
compar    lda (ptr),y          compare la commande entrée et 'xcat'
          cmp command-1,y
          bne nocmd            ce n'est pas pour nous
          iny
          cpy £4+1             commande 'xcat': 4 caractères
          bcc compar
          dey
          dey                  -> y = 3
          sty xlen
          lda £0
          sta syserr
          sta xcnum            n° commande externe
          lda £%00010000       nom de fichier autorisé
          sta pbits
          lda £%00000100       slot et drive autorisés
          sta pbits+1
versuite  lda xsuite           utile au relogeur seulement
          lda versuite+1       adresse à laquelle ProDOS devra
          sta xtrnaddr         nous rappeler après analyse
          lda versuite+2       des paramètres
          sta xtrnaddr+1
          clc                  retour à ProDOS sans erreur pour
          rts                  analyse des paramètres
nocmd     sec                  utile si notre commande est la seule
;                              ou la dernière pour signaler l'erreur
precmd    jmp xreturn          saut à la commande externe
;                              précédente

syntaxer  lda £$10             code de 'SYNTAX ERROR'
          jmp errout           affiche le message

; passons aux choses sérieuses...

xsuite    ldx £0
          lda output
          cmp £$C3             carte 80 colonnes active ?
          bne no80c            non
          dex                  oui -> carte80c = £$FF
no80c     stx carte80c
          ldy £0
          lda fbits            un nom de volume demandé ( fbits <> 0 ) ?
          beq pasvol           non

;                              oui (ptr) est toujours égal à (vpathl)
          lda (ptr),y          longueur du nom
          sta longueur         elle va servir
          cmp £3               longueur < 3 ?
          bcc syntaxer         oui -> erreur (le nom doit contenir
;                              au minimum 2 / et un caractère )
          tay
          lda (ptr),y          lit le dernier caractère
          cmp £$2F             '/' ?
          bne syntaxer         non
          ldy £1               oui -> lit le 1-er caractère
          lda (ptr),y
          cmp £$2F             '/' ?
          bne syntaxer         non
chslash   iny                  oui -> on cherche un '/' intermédiaire
          cpy longueur
          bcs nomok
          lda (ptr),y
          cmp £$2F             '/' ?
          beq syntaxer         oui -> erreur
          cmp £$2E             '.' ?
          bne cs2              non
cs1       cpy £2               oui -> en premier caractère ?
          beq syntaxer                oui : erreur
          bne chslash                 non : autorisé
cs2       cmp £$30             < '0'
          bcc syntaxer         non
          cmp £$3A             <= '9'  ( $3A = code de ':' )
          bcc cs1              oui -> vérifier si pas en 1-ère position
          cmp £$41             < 'A'
```

```
         bcc syntaxer         oui
         cmp £$5B             <= 'Z'  ( $5B = code de '[' )
         bcs syntaxer
         bcc chslash
nomok    jsr nomvol
         jmp go
pasvol   lda fbits+1          a-t-on fourni un paramètre s ou d ?
         bne calcunit         oui
         lda $BF9A            contient 0 si pas de prefix
         beq dernier          -> dernier lecteur utilisé
         jsr mli
         dc i1'$C7'           code de GET PREFIX
         dc i2'prefparm'
         jsr mli
         dc i1'$C6'           code de SET PREFIX
         dc i2'prefparm'
         bcc dernier          pas d'erreur ?
         ldx pile             hélas, si !
         txs
         jsr badcall
         jmp errout
dernier  lda devcnt-1         non -> on prend la dernière unité accédée
;                             (son numéro est en $BF30)
         bne stunit           toujours
calcunit lda vdriv            contient 0000 00DD avec DD=01 (drive 1)
;                                                  ou DD=10 (drive 2)
         and £$02             -> A = 0000 00D0 avec D=0 (drive 1)
;                                                ou D=1 (drive 2)
         asl a
         asl a                -> A = 0000 D000
         ora vslot            vslot contient 0000 0SSS
;                             -> A = 0000 DSSS
         asl a
         asl a
         asl a
         asl a                -> A = DSSS 0000
stunit   sta unite

;cette fois, c'est vraiment parti

go       tsx                  la pile va servir de zone de
         stx pile             stockage...sauvons le pointeur
         lda £0               lecture du bloc 00 02
         sta increm
         sta bloclu+1
         sta compteur
         sta buffer
         sta debascat
         dec debsscat         -> debascat = $FF , on devra
;                             souligner le 1-er nom de fichier
;                             qui est celui de la disquette
         lda himem+1          himem contient l'adresse du buffer
         sta buffer+1         brouillon de ProDOS
         lda £2               toujours pour le bloc 2
         sta lignes
         sta bloclu
         jsr litbloc
         lda devcnt-1         numéro du dernier lecteur accédé
         cmp £$B0             est-ce le ram disk 64k ? (s3, d2)
         beq prodos           si oui, inutile de vérifier les 3 premiers
;                             octets du bloc 2
         ldy £0               on lit les 3 premiers octets de ce bloc
         lda (himem),y        si <> 00 00 03 -> ce n'est pas un
         iny                  disque ProDOS ( les 2 premiers
         ora (himem),y        correspondent au chainage arrière:
         bne noprodos         nuls puisque ce bloc est le premier,
         iny                  le 03 normalement suivi de 00
         lda (himem),y        correspond au chainage avant: bloc
         cmp £3               suivant du catalogue )
         beq prodos
noprodos lda £5               -> 'I/O ERROR'
         jmp errout
prodos   ldy £$27             sauve le n° de bloc de la bitmap
         lda (himem),y        (octets $27 et $28 du bloc 2): il vaut
;                             presque toujours 00 06 sauf pour /ram: 00 03
         sta bitmap
         iny
         lda (himem),y
         sta bitmap+1
         iny
         lda (himem),y        nombre maxi de blocs du support
         sta blocmax          (octets $29 et $2A du bloc 2)
         iny
         lda (himem),y
         sta blocmax+1
         jsr crdo
         jsr datecat
         jsr crdo             on va imprimer la ligne "NOM..BLOCS...."
         jsr crdo
         ldy titres
titrebcl lda titres,y
         jsr outdo
         dey
         bne titrebcl
fintitre jsr crdo
         jsr crdo
         jmp analyseb         inutile de relire le bloc 2
blocsuiv jsr litbloc
analyseb lda kbd              arret avec CTRL-C ?
```

```
         and £$7F             meme si le CTRL-C date un peu ( clavier
         cmp £3               remis en attente avec kbstrobe
         bne continue         non
         jmp ctrlc            oui -> on écrit la dernière ligne
continue ldy compteur
         lda lowb,y           adresse relative de l'enregistrement
         clc
         adc himem            calcul adresse absolue de l'enreg.
         sta litbuf+1
         lda highb,y
         adc himem+1
         sta litbuf+2
         ldy £0
         jsr litbuf           lit le 1-er octet de l'enregistrement
;                             qui contient le type de sauvegarde
;                             du fichier et la longueur du nom:
;                             octet de la forme tttt llll
         beq suivant          $0 si fichier inexistant ou détruit
         and £$F0             -> A = tttt 0000
         cmp £$D0             est-ce un sous-catalogue ?
         bne autres           branche si autre que sous-catalogue
         lda compteur         c'est un nom de sous-catalogue
         pha                  -> on sauve le compteur et le
         lda bloclu+1         numéro du bloc
         pha
         lda bloclu
         pha
         lda £$FF             il faudra souligner
         sta debsscat
         ldy £$11             pour lire le n° du 1-er bloc du
         jmp nouvbloc         sous-catalogue
autres   jsr espaces          au retour, (Y) = 0
         jsr litbuf
         and £$0F             -> A = 0000 llll
         sta longueur
         tax
         iny                  (Y) = 1 : début du nom
affbcl1  jsr litbuf
         jsr outdo
         iny
         dex
         bne affbcl1
         bit debsscat         nom de cat ou sous-cat ?
         bmi alaligne         oui
         jsr infos            non
alaligne jsr crdo
         bit debsscat         on souligne ?
         bpl suivant          non
         inc debascat         -> debsscat = 0
         jsr espaces          oui
         ldx longueur
         lda £$2D             trait de soulignement
affbcl2  jsr outdo
         dex
         bne affbcl2
         jsr crdo
         inc increm
         inc increm
suivant  inc compteur         enregistrement suivant
         lda compteur
         cmp £$0D             déjà 13 fichiers affichés ?
         beq blocfini         oui
         jmp analyseb         non
blocfini lda buffer           on va lire les 2 octets de
         sta litbuf+1         chainage avant (octets numéros
         lda buffer+1         $2 et $3 du bloc) : c'est le
         sta litbuf+2         numéro du bloc suivant
         ldy £2
nouvbloc jsr litbuf
         sta bloclu
         iny
         jsr litbuf
         sta bloclu+1
         ora bloclu           si 00 00 -> le dernier bloc vient
         beq catfini          d'etre lu
         lda £0
         sta compteur
         jmp blocsuiv
catfini  tsx                  pointeur de pile égal à sa valeur de
         cpx pile             début ? ( <-> a-t-on tout dépilé ?)
         beq termine          oui -> catalogue principal terminé
         pla                  non -> on récupère le n° de bloc et
         sta bloclu           le compteur précédents
         pla
         sta bloclu+1
         pla
         sta compteur
         dec increm
         dec increm
         jsr litbloc          on relit le bloc précédent
         jmp suivant
ctrlc    ldx £5               en cas de CTRL-C on imprime '.....'
         lda £$2E             '.'
asuivre  jsr outdo
         dex
         bne asuivre
         ldx pile             on laisse les lieux comme on
         txs                  les a trouvés
termine  jsr crdo
```

```
         jsr crdo
         lda bitmap          il ne reste qu'à afficher la ligne
         sta bloclu                  'blocs libres...'
         lda bitmap+1
         sta bloclu+1
         jsr litbloc         lit la bitmap
         lda himem           on va explorer 512 octets du buffer
         sta litbuf+1
         lda himem+1
         sta litbuf+2
         lda £0
         sta nbloc           nbloc,nbloc+1 contiendra le nombre
                             de blocs libres
         sta nbloc+1
         jsr litbitmp        on va lire la 1-ère moitié de la bitmap
         inc litbuf+2
         jsr litbitmp        2-ème moitié
         ldy libres
t1       lda libres,y        mention "blocs libres: "
         jsr outdo
         dey
         bne t1
         jsr impradr         imprime nombre de blocs libres
         lda blocmax         calcule blocs occupés...
         sec
         sbc nbloc
         sta nbloc
         lda blocmax+1
         sbc nbloc+1
         sta nbloc+1
         ldy occupes         mention "occupés: "
t2       lda occupes,y
         jsr outdo
         dey
         bne t2
         jsr impradr
         lda blocmax
         sta nbloc
         lda blocmax+1
         sta nbloc+1
         ldy total           mention "total blocs: "
t3       lda total,y
         jsr outdo
         dey
         bne t3
         jsr impradr         imprime total blocs
         jsr crdo
         bit kbstrobe
         clc                 pour ProDOS : pas d'erreur
         rts

; ----------------------------------------------------------------
;                       SOUS PROGRAMMES
; ----------------------------------------------------------------

; lecture d'un enregistrement du bloc lu
; ----------------------------------------

litbuf   lda $8000,y      adresse non significative, modifiée
         rts               selon les besoins

; imprime les espaces de début de ligne
; ---------------------------------------

espaces  ldy increm
         beq esprts
         cpy incmax          y < incmax ?
         bcc espacbis        oui
         ldy incmax          non -> y = incmax pour limiter le
;                            décalage à une valeur raisonnable
espacbis lda £$A0
espbcl   jsr outdo
         dey
         bne espbcl
esprts   rts

; imprime un retour chariot
; ---------------------------

crdo     inc lignes
         lda lignes
         cmp £$3B            déjà 60 lignes affichées ?
         bcc memepage        non
         bit carte80c        oui -> 80 col active ?
         bmi noformf                oui
         bit debsscat               non -> nom de ss-cat ?
         bmi memepage                      oui
         lda £$0C                          non -> saut de page
         jsr outdo
noformf  lda £0
         sta lignes
memepage lda £$FF
         sta htab
         lda £$8D
         jsr outdo
         rts

; imprime le caractère (A) ou un espace et incrémente htab
; ----------------------------------------------------------

outspace lda £$A0

outdo    pha
         lda kbd             une touche enfoncée ?
         bpl out             non
         bit kbstrobe        oui
         cmp £$83            ctrl-c ?
         beq out             oui
attends  bit kbd             non -> attend une nouvelle touche
         bpl attends
         bit kbstrobe
out      pla
         jsr $DB5C
         inc htab
         rts

; imprime le type, le nombre de blocs et la date
; ------------------------------------------------

infos    ldx htab            le type commence en colonne 41
         lda £$A0
il       jsr outdo
         inx
         cpx £$28            $28 = 40
         bcc il
         jsr type
         jsr blocs
         jsr date
         rts

; analyse 256 octets de la bitmap
;------------------------------------

litbitmp lda £$0
         tay
bmp1     jsr litbuf
         beq bmp4            octet = 00 -> on passe au suivant
         sta temp
         ldx £8              8 bits à ajouter
         lda nbloc
bmp2     lsr temp            le bit b0 tombe danc c...
         adc £0              ... et est ajouté à A
         bcc bmp3
         inc nbloc+1
bmp3     dex                 bit suivant ?
         bne bmp2            oui
         sta nbloc           non
bmp4     iny
         bne bmp1            octet suivant
         rts

; lit et affiche le nombre de blocs occupés par le fichier
; ------------------------------------------------------------

blocs    jsr outspace
         ldy £$13            nombre de blocs du fichier dans les octets
         jsr litbuf          $13 et $14 de l'enregistrement
         sta nbloc
         iny
         jsr litbuf
         sta nbloc+1

;suivi immédiatement par impradr

; imprime un décimal de 5 chiffres maxi cadré à droite
; ------------------------------------------------------

; ce nombre est contenu dans nbloc et nbloc+1

; si le décimal est:
;         inférieur à 10 -> 3 espaces        10 = $ 00 0A
;   compris entre 10 et 100 -> 2    "       100 = $ 00 64
; compris entre 100 et 1000 -> 1    "      1000 = $ 03 E8
;         supérieur à 1000 -> 0     "
; + 1 espace pour le séparer de ce qui précède

impradr  ldy £0
         lda nbloc+1
         cmp £3              nombre < $03 00 ?
         bcc b1              oui
         bne impbloc         nombre >= $04 00
         lda nbloc           non
         cmp £$E8            nombre supérieur à $03 E8 ?
         bcs impbloc         oui ( >= 1000 )
         bcc ilsp            non ( < 1000 )
b1       lda nbloc+1
         bne ilsp            nombre supérieur à 256
         lda nbloc
         cmp £$64            supérieur à 100 ?
         bcs ilsp            oui
         cmp £$0A            non -> supérieur à 10 ?
         bcs i2sp                   oui
         iny                        non
i2sp     iny
ilsp     iny
impbloc  iny
         jsr espacbis
         lda nbloc+1
         ldx nbloc
```

```
        jsr linprint
        jsr outspace
        rts

; imprime le type du fichier:
; --------------------------
;       soit par son abréviation
;       soit sous la forme $xx

; beaucoup de types possèdent une abréviation qui dépend
; du logiciel utilisant les fichiers. La liste des codes
; est stockée à l'envers à partir de $B989 et est suivie
; de la liste des codes.

type     jsr outspace
         ldy £$10              le type de fichier est dans
         jsr litbuf            l'octet $10
         ldx £$0D              14abréviations différentes
         ldy £0
typebcl  cmp typnum,x          (A) = code lu ?
         beq afftype           oui
         iny                   non
         iny
         iny
         dex                   au suivant ?
         bpl typebcl           oui
         tax                   non ( A contient aussi le type )
         lda £$24              code de '$'
         jsr outdo
         txa
         jsr prbyte            imprime le type en 2 chiffres hexa
         jsr outspace
         rts
afftype  ldx £3                3 caractères à imprimer
afftbcl  lda typtable,y        Y contient déjà l'offset du type
         jsr outdo             à afficher
         iny
         dex
         bne afftbcl
         jsr outspace
         rts

; imprime la date d'impression du catalogue :
; ----------------------------------------------

datecat  jsr mli               demande à ProDOS de lire la date, elle
;                              sera stockée dans datesys, datesys+1
         dc il'$82'            code de GET TIME
         dc i2'0'              uniquement pour faire joli !!!
;                              GET TIME n'utilise pas de paramètre
;                              mais une adresse est nécessaire après
;                              le code de l'appel
         lda £<datesys-$21     la routine date utilise Y=$21 et Y=$22
         sta litbuf+1
         lda £>datesys-$21
         sta litbuf+2
         ldy msgdate
affmsg   lda msgdate,Y
         jsr outdo
         dey
         bne affmsg

; suivi immédiatement de date
; affichage de la date de modification du fichier
; -----------------------------------------------

; la date est stockée dans les octets $21 et $22 de
; l'enregistrement avec le format :

;               octet $22     octet $21
;               aaaa aaam     mmmj jjjj

date     jsr outspace
         ldy £$22
         jsr litbuf
         beq sansdate
         cmp £$C8              octet $22 >= £$ C8 ? ( <-> année >= 100)
         bcs sansdate
         lsr a                 c = bit b0: fait partie du mois
         sta aa
         dey
         jsr litbuf            lecture octet $21
         beq sansdate
         sta jj
         ror a                 on fait entrer la retenue à gauche
         lsr a                 et on garde b7..b4 par décalages
         lsr a
         lsr a
         lsr a
         cmp £$0D              mois > 12 ?
         bcs sansdate          oui -> sans date
         sta mm
         lda jj
         and £$1F              jour = b4..b0
         cmp £$20              jour >= 32 ?
         bcs sansdate          oui -> sans date
         cmp £$0A              non -> jour < 10 ?
         bcs datel                   non
         pha                         oui
         jsr outspace                    -> un espace
         pla
datel    jsr imprdate          imprime le jour
         lda £$2D              code de "-"
         jsr outdo
         lda mm                calcul adresse relative du 1-er
         asl a                 caractère du mois: mm * 3
         adc mm
         tay
         ldx £$3               3 caractères
ms       lda mois-3,y          table des noms des mois de ProDOS
         jsr outdo
         iny
         dex
         bne ms
         lda £$2D              '-'
         jsr outdo
         lda aa                imprime l'année
         jsr imprdate
         rts
sansdate ldx £$24              début du message '(NO DATE)'
affsd    lda mois,x
         jsr outdo
         inx
         cpx £$2D
         bcc affsd
         rts

imprdate tax                   imprime le contenu de A (MSB)
         lda £$0               et X (LSB) en décimal
         jsr linprint
         rts

; lecture d'un bloc
; -----------------

litbloc  jsr mli
         dc il'$80'            code de READ BLOCK
         dc i2'readparams'     adresse table des paramètres
         bcc okl               c = 0 -> pas d'erreur
         ldx plle              sortie en catastrophe
         txs
         jsr badcall           traitement erreur
         jsr errout
okl      rts

; recherche des volumes en ligne
; ------------------------------

; On passe en revue les lecteurs en ligne dans l'ordre
; croissant des slots et drives jusqu'à ce que l'on
; ait trouvé le volume cherché.

nomvol   dec longueur          longueur du nom sans les 2 /
         dec longueur
         lda himem             le meme buffer recevra le nom
         sta bufferol          de chaque volume trouvé par
         sta litbuf+1          ON LINE
         lda himem+1
         sta bufferol+1
         sta litbuf+2
         inc ptr               avance ptr d'un cran pour sauter
         bne nomvoll           le slash
         inc ptr+1
nomvoll  ldx devcnt            nombre d'unités connectées
onlinebcl lda devcnt+1,x       le nombre trouvé est de la forme:
         and £$F0              dsss tttt où tttt désigne le
;                              type de support (disque souple, dur
;                              ram...) -> on élimine tttt
         sta unitol
         jsr mli
         dc il'$C5'            code de ON LINE
         dc i2'onlineparams'   table des paramètres
         bcs unitsuiv          erreur quelconque -> au suivant
         ldy £0
         jsr litbuf            lit le 1-er octet du buffer qui
         sta temp              contient: dsss llll où llll est
         and £$0F              la longueur du nom du volume
         cmp longueur          2 longueurs égales ?
         bne unitsuiv          non
         tay                   oui -> Y = longueur
compnom  jsr litbuf            compare les noms de volumes
         cmp (ptr),y           demandé et lu par ON LINE
         bne unitsuiv          ils sont différents
         dey
         bne compnom           fini ? non
         lda temp              oui
         and £$F0              A = dsss 0000
         sta unite
         rts                   mission accomplie
unitsuiv dex                   unité suivante ?
         bpl onlinebcl         oui
         lda £6                non -> on n'a rien trouvé:
;                              code de 'PATH NOT FOUND'
         jmp errout            insulte

; ------------------------------------------------------------
;                bulle nécessaire au relogeur
; ------------------------------------------------------------
```

```
rien      ds 1

; ---------------------------------------------------------------
;                        tables et textes
; ---------------------------------------------------------------

; paramètres de READ BLOCKS
; -------------------------

readparams anop
          dc i1'3'               3 paramètres
unite     dc i1'$50'             lecteur s5, d1
buffer    ds 2                   on y stockera l'adresse du buffer
;                                brouillon de ProDOS
bloclu    ds 2                   numéro du bloc à lire

; paramètres de ON-LINE:
; ---------------------

onlineparams anop
          dc i1'2'               2 paramètres
unitol    ds 1                   numéro d'unité
bufferol  ds 2                   meme buffer que pour READ

; paramètres de GET-PREFIX et SET-PREFIX:
; ---------------------------------------

prefparm anop
          dc i1'1'               1 seul paramètre
          dc i2'$200'            le buffer d'entrée

; adresses relatives des enregistrements d'un catalogue
; -----------------------------------------------------
; ou d'un sous-catalogue par rapport au début du buffer
; -----------------------------------------------------

lowb      dc i1'$04,$2B,$52,$79,$A0,$C7,$EE'
          dc i1'$15,$3C,$63,$8A,$B1,$D8'

highb     dc i1'0,0,0,0,0,0,0'
          dc i1'1,1,1,1,1,1'

; ligne de titres du catalogue
; ----------------------------

titres    dc i1'124'
          dc c'----------------------------------------'
          dc c'----------------------'
          dc i1'$8D'
          dc c'EIFIDOM  SCOLB  EPYT'
          dc c'                                      MON'

; textes et constantes
; --------------------

command   dc c'XCAT'
nonactif  dc i1'25,13,13,7'
          dc c'** FICTA NON SODORP **'
libres    dc i1'15'
          dc c' : SERBIL SCOLB'
occupes   dc i1'12'
          dc c' : SEPUCCO  '
total     dc i1'10'
          dc c' : LATOT  '
msgdate   dc i1'$17'
          dc c' : EL EMIRPMI EUGOLATAC'
incmax    dc i1'24'              décalage maximum de 24 caractères

; ---------------------------------------------------------------

fin       gequ *

          END
```

# Récapitulation XCAT

*Après avoir saisi ce code sous moniteur, vous le sauvegarderez par :*

BSAVE XCAT, A$2000, L1685

```
2000:AD 00 BF C9 4C F0 0D AC
2008:39 26 B9 39 26 20 D6 23
2010:88 D0 F7 60 AD 4D BE F0
2018:05 A9 15 4C 09 BE AD 08
2020:BE 8D 44 21 AD 07 BE 8D
2028:43 21 A9 06 20 F5 BE C9
2030:0C D0 03 4C 09 BE 8D E2
2038:20 85 43 8D 08 BE A9 21
2040:85 3D A9 93 85 3E A9 26
2048:85 3F A0 00 84 42 84 3C
2050:8C 07 BE 20 2C FE 20 5A
2058:20 60 AD E2 20 85 3B A9
2060:00 85 3A A2 00 A1 3A F0
2068:50 20 8C F8 A4 2F C0 02
2070:D0 2C B1 3A C9 BF D0 19
2078:88 B1 3A C9 00 D0 12 88
2080:B1 3A C9 20 D0 0B A5 3A
2088:18 69 03 85 3A 90 02 E6
2090:3B A5 3A 85 FC A5 3B 85
2098:FD A4 2F 20 C8 20 A5 2F
20A0:38 65 3A 85 3A A5 3B 69
20A8:00 85 3B 38 ED E2 20 A6
20B0:3A 20 BA 20 B0 03 4C 63
20B8:20 60 C9 05 90 06 D0 06
20C0:E0 94 B0 02 18 60 38 60
20C8:88 B1 FC AA C8 B1 FC 38
20D0:E9 21 90 0C 20 BA 20 B0
20D8:07 6D E2 20 91 FC 18 60
20E0:38 60 00 00 00 00 00 00
20E8:00 00 00 00 00 00 00 00
20F0:00 00 00 00 00 00 00 00
20F8:00 00 00 00 00 00 00 00
2100:D8 AD 6C BE 85 FE AD 6D
2108:BE 85 FF A0 01 B1 FE D9
2110:34 26 D0 2D C8 C0 05 90
2118:F4 88 88 8C 52 BE A9 00
2120:8D 0F BF 8D 53 BE A9 10
2128:8D 54 BE A9 04 8D 55 BE
2130:AD 4A 21 AD 31 21 8D 50
2138:BE AD 32 21 8D 51 BE 18
2140:60 38 4C 9E BE A9 10 4C
2148:09 BE A2 00 AD 31 BE C9
2150:C3 D0 01 CA 86 E3 A0 00
2158:AD 56 BE F0 44 B1 FE 85
2160:1A C9 03 90 E0 A8 B1 FE
2168:C9 2F D0 D9 A0 01 B1 FE
2170:C9 2F D0 D1 C8 C4 1A B0
2178:22 B1 FE C9 2F F0 C6 C9
2180:2E D0 06 C0 02 F0 BE D0
2188:EB C9 30 90 B8 C9 3A 90
2190:F2 C9 41 90 B0 C9 5B B0
2198:AC 90 D9 20 3B 25 4C D8
21A0:21 AD 57 BE D0 21 AD 9A
21A8:BF F0 17 20 00 BF C7 9B
21B0:25 20 00 BF C6 9B 25 90
21B8:09 A6 1E 9A 20 8B BE 4C
21C0:09 BE AD 30 BF D0 0E AD
21C8:62 BE 29 02 0A 0A 0D 61
21D0:BE 0A 0A 0A 0A 8D 92 25
21D8:BA 86 1E A9 00 85 1D 8D
21E0:96 25 85 1C 8D 93 25 85
21E8:1B C6 1B A5 74 8D 94 25
21F0:A9 02 85 EF 8D 95 25 20
21F8:29 25 AD 30 BF C9 B0 F0
2200:15 A0 00 B1 73 C8 11 73
2208:D0 07 C8 B1 73 C9 03 F0
2210:05 A9 08 4C 09 BE A0 27
2218:B1 73 85 EB C8 B1 73 85
2220:EC C8 B1 73 85 ED C8 B1
2228:73 85 EE 20 B1 23 20 9E
2230:24 20 B1 23 20 B1 23 AC
2238:B8 25 B9 B8 25 20 D6 23
2240:88 D0 F7 20 B1 23 20 B1
2248:23 4C 4F 22 20 29 25 AD
2250:00 C0 29 7F C9 03 D0 03
2258:4C 15 23 A4 1C B9 9E 25
2260:18 65 73 8D 99 23 B9 AB
2268:25 65 74 8D 9A 23 A0 00
2270:20 98 23 F0 54 29 F0 C9
2278:D0 D0 14 A5 1C 48 AD 96
2280:25 48 AD 95 25 48 A9 FF
2288:85 1B A0 11 4C E2 22 20
2290:9C 23 20 98 23 29 0F 85
2298:1A AA C8 20 98 23 20 D6
22A0:23 C8 CA D0 F6 24 1B 30
22A8:03 20 F2 23 20 B1 23 24
22B0:1B 10 16 E6 1B 20 9C 23
22B8:A6 1A A9 2D 20 D6 23 CA
22C0:D0 FA 20 B1 23 E6 1D E6
22C8:1D E6 1C A5 1C C9 0D F0
22D0:03 4C 4F 22 AD 93 25 8D
22D8:99 23 AD 94 25 8D 9A 23
22E0:A0 02 20 98 23 8D 95 25
22E8:C8 20 98 23 8D 96 25 0D
22F0:95 25 F0 07 A9 00 85 1C
22F8:4C 4C 22 BA E4 1E F0 22
2300:68 8D 95 25 68 8D 96 25
2308:68 85 1C C6 1D C6 1D 20
2310:29 25 4C C9 22 A2 05 A9
2318:2E 20 D6 23 CA D0 FA A6
2320:1E 9A 20 B1 23 20 B1 23
2328:A5 EB 8D 95 25 A5 EC 8D
2330:96 25 20 29 25 A5 73 8D
2338:99 23 A5 74 8D 9A 23 A9
2340:00 85 06 85 07 20 08 24
2348:EE 9A 23 20 08 24 AC 53
2350:26 B9 53 26 20 D6 23 88
2358:D0 F7 20 37 24 A5 ED 38
2360:E5 06 85 06 A5 EE E5 07
2368:85 07 AC 63 26 B9 63 26
2370:20 D6 23 88 D0 F7 20 37
```

*Suite page 37...*

# Des filets en WPL

## pour ImageWriter & DMP

Robert Coustal

Les imprimantes ImageWriter et DMP possèdent un grand nombre de possibilités qui ne sont généralement pas exploitées par l'utilisateur. En effet, il faut lui envoyer des codes de commandes relativement complexes qui nécessitent souvent un programme particulier peu compatible avec d'autres logiciels, et dont les effets semblent quelquefois aléatoires.

Heureusement le traitement de texte AppleWriter permet d'envoyer ces codes à l'ImageWriter et son langage de programmation le WPL (Word Processing Language) peut se charger de calculer les paramètres nécessaires aux commandes à envoyer.

Le programme *WPL.FILET* se charge de tracer des filets graphiques décoratifs et centrés, pour encadrer un titre ou bien pour séparer des paragraphes, sans quitter le texte en cours, en incluant les caractères de commande dans le texte. Il fonctionne avec Applewriter, version DOS 3.3. ou ProDOS, et avec une imprimante de la série ImageWriter.

## Le principe

Sur l'ImageWriter le code *ESCAPE-Vnnnnc* permet de tracer une ligne répétant *nnnn* fois l'image du code binaire ASCII du caractère *c*. Le problème consiste donc, une fois choisi le caractère *c* à calculer le nombre *nnnn* pour obtenir la longueur de ligne voulue, et à ajouter à ce code un certain nombre d'espace pour obtenir une justification au centre correspondant à la longueur du filet à tracer et non à la longueur du code seul. Pour obtenir deux filets encadrant une ligne de texte, il faut de plus connaître le nombre de caractères de celle-ci. *WPL.FILET* se chargera de ces calculs.

## Fonctionnement

*WPL.FILET*

- insère un marqueur à l'endroit du texte où vous souhaitez opérer,
- compte le nombre de caractères du titre que vous voulez placer (s'il n'y a qu'un seul filet à tracer sa longueur est demandée à l'utilisateur),
- multiplie ensuite ce nombre par 8 pour connaître le nombres de points qui correspond à la taille de la ligne,
- écrit ce nombre en respectant le format de 4 caractères (avec des zéros en tête si nécessaire),
- ajoute ensuite des espaces pour obtenir une longueur de ligne correcte pour la justification au centre.

Si votre texte ou votre ligne compte moins de 8 caractères (c'est-à-dire la longueur minimum du code de commande), la ligne sera allongée automatiquement. Si vous désirez une série de lignes de longueur décroissantes, le programme se chargera de calculer une ligne plus courte de 4 caractères au-dessous de la précédente.

## Utilisation

Deux possibilités :

- taper *CTRL-P DOWPL.FILET*,
- mettre le programme "sous la pomme" en utilisant le glossaire. Il faut faire *CTRL-G* puis *?* et choisir un caractère pour faire exécuter le programme (par exemple '_'), tapez alors

*_CTRL-PDOWPL.FILET>*

suivi de Return (> dans Applewriter ProDOS ou *$* dans Applewriter DOS 3.3.). Chaque fois que vous voudrez utiliser le programme tapez seulement '_'.

Le programme une fois lancé vous demande de taper votre titre ou <Return> si vous voulez seulement une ligne. Il demande ensuite le code du filet, c'est-à-dire le caractère dont le code ASCII dessinera le filet (chaque bit à 1 de ce code tracera une ligne horizontale). Vous pouvez essayer différent codes, (les caractères = V U u g m par exemple, donnent de jolis filets), le programme vous fournit quelques exemples et vous pouvez une fois de retour au texte changer le caractère s'il ne vous convient pas. Si vous avez choisi d'encadrer un texte, <Return> seul en réponse à la demande du code du filet du bas conservera le même code que pour celui du haut.

Les titres et les filets sont insérés à l'emplacement du curseur dans

le texte et écrivent le code .JT d'Applewriter pour remettre le texte en justification totale. Vous pouvez changer .JT dans le programme par .JG si vous préférez la justification à gauche.

Il existe toutefois certaines limitations : on ne peut pas choisir des codes ASCII inférieurs à 128 et le code 255 du fait d'AppleWriter ; de plus, WPL.FILET intercepte les caractères `< > ?` et `Return` (`CTRL-H` et le paramètre du souligné ne sont pas conseillés).

## Saisie

Un programme WPL ne peut pas dépasser 2048 caractères. WPL.FILET étant un peu long, on peut supprimer les espaces inutiles en tête des lignes (en en conservant toutefois un avant chaque commande), et les exemples de codes au début. Sauver ensuite en faisant *CTRL-S WPL.FILET*.

D'autre part, si votre Applewriter ProDOS est patché pour ne pas compter les caractères Escape en justification totale il faut changer tous les PSX-7 en PSX-5 et les PSZ8 en PSZ6 pour obtenir un centrage correct.

CREATION DE FILETS GRAPHIQUES AVEC APPLEWRITER SUR IMAGEWRITER

# Programme WPL.FILET

*À saisir à l'aide d'AppleWriter. Pour en simplifier le comptage, des espaces ont été replacés par des puces (•).*

```
DEBUT PND
      PPR1
      PPR  FILETS AUTOMATIQUES (R.Coustal/87)
      PPR  ==================================
      PPR
      PPRQuelques codes:
      PPR
      PPRlignes:•fines•••••larges••••mixtes
      PPR
      PPR  1:•••••à`•••••••••••¨ùx
      PPR  2:•<A><B><D><P>•••gnF••••<C>8O^bz
      PPR  3:••<R><I><T>(•••••°•••=:'76ITWZjum
      PPR  4:••••••*&)••••••••••••••••••U5+
      PPR
      PINTitre à encadrer (return filet seul):=$A
      PCS<$A<<
      PGOLI1
      PGOTITRE
LI1   PINLongueur en caractères:=$B
      PSX$B
      PAS(X)=$B
      PCS<$B<0<
      PGOLI1
      PSRPLACE
FILET PINCode du filet:=$C
      PSRFILTR
      PCS<$C<<
      PGOFILET
      PGOVERI2
VERI1 PSX+1
      PAS(X)=$B
VERI2 PSY(X)
      PSZ8
BCL   PSZ-1
      PGODEC1
      PGOLI2
DEC1  PSY-1
      PGOBCL
      PGOVERI1
LI2   PSRCALC
      PSRBLANC
      PINune autre ligne plus courte? (O/N):=$A
      PCS<$A<o<
      PASO=$A
      PCS<$A<O<
      PGOAUTRE
      PQT
AUTRE F<<^^<
      O?
      HHDXB
      PSX$B
      PSX-4
      PAS(X)=$B
      PGOVERI2
TITRE PSRPLACE
      F<<$A`<
      O?
      PSX0
COMPT DHH
      F<?<?_<A
      B
      F<`<<
      O?
      D
INC   PSX+1
      PGOBCL0
BCL0  F<?_<<
```

```
       O?
       PGOINC
TOTAL  PSX-1
       B
VERI3  PAS $A =$A
       PSX+2
       PSY(X)
       PSZ8
BCL1   PSZ-1
       PGODEC2
       PGOECRIT
DEC2   PSY-1
       PGOBCL1
       PGOVERI3
ECRIT  F<^^<^^$A^^<
       O?
       PSRCALC
HAUT   PINCode du filet haut:=$C
       PSRFILTR
       PCS<$C<<
       PGOHAUT
       B
       PSRBLANC
BAS    PINCode du filet bas:=$D
       PCS<$D<<
       PGOBAS2
       PAS$D=$C
       PSRFILTR
       PCS<$C<<
       PGOBAS
BAS2   PSRCODE
       PQT
```

DOS 3.3
ProDOS

```
PLACE  F<<>.JC^^.JT><
       O?
       B
       PRT
CALC   PSY(X)
       PSZ0
*8     PSZ+8
       PSY-1
       PGO*8
       PRT
BLANC  PSX-7
       PAS=$A
BCL2   PAS$A =$A
       PSX-1
       PGOBCL2
CODE   F<^^<>2V>000(Z)$C><
       O?
       DHHHHHHXD
       F<><$A><
       O?
       PRT
FILTR  PCS<$C</<
       PRT
       PCS/$C/?/
       PAS=$C
       PCS/$C/>/
       PAS=$C
       PRT
```

//e
//e+
//c
]\[gs

***1- mettre un CTRL-L (taper au clavier CTRL-V CTRL-L CTRL-V)***
***2- mettre Escape (taper CTRL-V ESC CTRL-V)***

*...Suite de la page 34*

```
2378:24 A5 ED 85 06 A5 EE 85
2380:07 AC 70 26 B9 70 26 20
2388:D6 23 88 D0 F7 20 37 24
2390:20 B1 23 2C 10 C0 18 60
2398:B9 00 80 60 A4 1D F0 10
23A0:CC 93 26 90 03 AC 93 26
23A8:A9 A0 20 D6 23 88 D0 FA
23B0:60 E6 EF A5 EF C9 3D 90
23B8:11 24 E3 30 09 24 1B 30
23C0:09 A9 0C 20 D6 23 A9 00
23C8:85 EF A9 FF 85 1F A9 8D
23D0:20 D6 23 60 A9 A0 48 AD
23D8:00 C0 10 0F 2C 10 C0 C9
23E0:83 F0 08 2C 00 C0 10 FB
23E8:2C 10 C0 68 20 5C DB E6
23F0:1F 60 A6 1F A9 A0 20 D6
23F8:23 E8 E0 28 90 F8 20 69
2400:24 20 27 24 20 BA 24 60
2408:A9 00 A8 20 98 23 F0 13
2410:85 08 A2 08 A5 06 46 08
2418:69 00 90 02 E6 07 CA D0
2420:F5 85 06 C8 D0 E5 60 20
2428:D4 23 A0 13 20 98 23 85
2430:06 C8 20 98 23 85 07 A0
2438:00 A5 07 C9 03 90 0A D0
2440:19 A5 06 C9 E8 B0 13 90
2448:10 A5 07 D0 0C A5 06 C9
2450:64 B0 06 C9 0A B0 01 C8
2458:C8 C8 C8 20 A8 23 A5 07
2460:A6 06 20 24 ED 20 D4 23
2468:60 20 D4 23 A0 10 20 98
2470:23 A2 0D A0 00 DD 89 B9
2478:F0 14 C8 C8 C8 CA 10 F5
2480:AA A9 24 20 D6 23 8A 20
2488:DA FD 20 D4 23 60 A2 03
2490:B9 97 B9 20 D6 23 C8 CA
2498:D0 F6 20 D4 23 60 20 00
24A0:BF 82 00 00 A9 6F 8D 99
24A8:23 A9 BF 8D 9A 23 AC 7B
24B0:26 B9 7B 26 20 D6 23 88
24B8:D0 F7 20 D4 23 A0 22 20
24C0:98 23 F0 50 C9 C8 B0 4C
24C8:4A 85 F9 88 20 98 23 F0
24D0:43 85 FB 6A 4A 4A 4A 4A
24D8:C9 0D B0 38 85 FA A5 FB
24E0:29 1F C9 20 B0 2E C9 0A
24E8:B0 05 48 20 D4 23 68 20
24F0:22 25 A9 2D 20 D6 23 A5
24F8:FA 0A 65 FA A8 A2 03 B9
2500:BE B9 20 D6 23 C8 CA D0
2508:F6 A9 2D 20 D6 23 A5 F9
2510:20 22 25 60 A2 24 BD C1
2518:B9 20 D6 23 E8 E0 2D 90
2520:F5 60 AA A9 00 20 24 ED
2528:60 20 00 BF 80 91 25 90
2530:09 A6 1E 9A 20 8B BE 20
2538:09 BE 60 C6 1A C6 1A A5
2540:73 8D 99 25 8D 99 23 A5
2548:74 8D 9A 25 8D 9A 23 E6
2550:FE D0 02 E6 FF AE 31 BF
2558:BD 32 BF 29 F0 8D 98 25
2560:20 00 BF C5 97 25 B0 20
2568:A0 00 20 98 23 85 08 29
2570:0F C5 1A D0 13 A8 20 98
2578:23 D1 FE D0 0B 88 D0 F6
2580:A5 08 29 F0 8D 92 25 60
2588:CA 10 CD A9 06 4C 09 BE
2590:00 03 50 00 00 00 00 02
2598:00 00 00 01 00 02 04 2B
25A0:52 79 A0 C7 EE 15 3C 63
25A8:8A B1 D8 00 00 00 00 00
25B0:00 00 01 01 01 01 01 01
25B8:7C 2D 2D 2D 2D 2D 2D 2D
25C0:2D 2D 2D 2D 2D 2D 2D 2D
25C8:2D 2D 2D 2D 2D 2D 2D 2D
25D0:2D 2D 2D 2D 2D 2D 2D 2D
25D8:2D 2D 2D 2D 2D 2D 2D 2D
25E0:2D 2D 2D 2D 2D 2D 2D 2D
25E8:2D 2D 2D 2D 2D 2D 2D 2D
25F0:2D 2D 2D 2D 2D 2D 2D 2D
25F8:8D 45 49 46 49 44 4F 4D
2600:20 20 53 43 4F 4C 42 20
2608:20 45 50 59 54 20 20 20
2610:20 20 20 20 20 20 20 20
2618:20 20 20 20 20 20 20 20
2620:20 20 20 20 20 20 20 20
2628:20 20 20 20 20 20 20 20
2630:20 20 4D 4F 4E 58 43 41
2638:54 19 0D 0D 07 2A 2A 20
2640:46 49 43 54 41 20 4E 4F
2648:4E 20 53 4F 44 4F 52 50
2650:20 2A 2A 0F 20 3A 20 53
2658:45 52 42 49 4C 20 53 43
2660:4F 4C 42 0C 20 3A 20 53
2668:45 50 55 43 43 4F 20 20
2670:0A 20 3A 20 4C 41 54 4F
2678:54 20 20 17 20 3A 20 45
2680:4C 20 45 4D 49 52 50 4D
2688:49 20 45 55 47 4F 4C 41
2690:54 41 43 18 60 C9
```

# Les disquettes Pom's

Pour éviter les saisies fastidieuses et pas toujours fiables, Pom's met à votre disposition des disquettes d'accompagnement qui regroupent l'ensemble des programmes de la revue.

## Apple //

Pour les Apple //, deux types de disquettes :

- 140Ko, 5,25 pouces au prix de 60,00 F, fichiers en format DOS 3.3 au recto, en format ProDOS au verso ;
- 800Ko, 3,5 pouces pour Unidisk, au prix de 80,00 F, fichiers en format ProDOS seulement.

Sur la 800Ko et la 140Ko face ProDOS (toutes deux nommées /POMS32), nous vous conseillons d'installer les fichiers 'ProDOS' et 'Basic.System', ainsi la disquette sera *bootable.*

Sur la liste ci-contre, les fichiers sont repérés ainsi :

D : face DOS 3.3 de la 140Ko ;
P : face ProDOS de la 140Ko ;
8 : disquette 800Ko.

## Macintosh

La liste ci-contre donne l'ensemble des fichiers de la disquette Mac, fichiers utilisables sur tous les types de Macintosh.

## Fichiers Apple //

| | | | | |
|---|---|---|---|---|
| | P | 8 | TRANSFORMEUR | Programme (RUN ou -) |
| | P | 8 | HGR.SUPHGR.C | Utilitaire pour Transformeur |
| | P | 8 | HGR.SUPHGR.S | Source en format TEXT |
| D | | 8 | BIGTEXT | Source en format TEXT |
| D | | 8 | BIGPGM | Source en format TEXT |
| D | | 8 | BIGSAVE | Source en format TEXT |
| D | | 8 | BIGPTR | Table |
| D | | 8 | ECR.VIRT | Code issu des 3 sources et de la table |
| D | | 8 | VIRPUZZLE | Pgm de création de ECR-VIRT (RUN) |
| D | | 8 | VIRCEL | Pgm de démonstration (RUN) |
| D | | 8 | VIDEO | Écran de démonstration pour VIRCEL |
| | P | 8 | XCATS | Source en format TEXT |
| | P | 8 | XCAT | Objet (BRUN ou -) |
| D | P | 8 | WPL.FILET | WPL à exécuter depuis AppleWriter |
| | P | 8 | COPY | Objet (BRUN ou -) |
| | P | 8 | COPY.S | Source en format TEXT |
| | P | 8 | COPIE.TF | Pgm Basic (RUN ou -) |
| | P | 8 | RECURSIVE.TEXT | Ces trois programmes Pascal |
| | P | 8 | U.GESTABL.TEXT | sont à convertir à l'aide |
| | P | 8 | P.REPERT.TEXT | d'Universal File Conversion |
| D | P | 8 | DETECT | Pgm Basic (RUN) |
| D | | | RECURSIVE | Ces trois progammes Pascal |
| D | | | U.GESTABL | sont à convertir à l'aide |
| D | | | P.REPERT | de Basic-Pascal |
| D | | | BASIC.PASCAL | Pgm de conversion DOS/PASCAL (RUN) |
| D | | | BASPAC.OBJ1 | Utilitaire pour BASIC.PASCAL |

## Fichiers Macintosh

| | |
|---|---|
| Dossier Système | Contient 'fuseaux' installé et quelques éléments vitaux |
| Font/DA Mover | Pour installer 'fuseaux' |
| Mots_Croisés | Application du même nom |
| Grille1 | Le problème posé dans ce numéro, page 73 |
| mots_croises.pas | Source de l'application 'Mots_Croisés' |
| Accessoire 'fuseaux' | Fichier Font/DA Mover qui contient 'fuseaux' |
| décalage.Asm | Source MDS 68000 de 'fuseaux' |
| mABcro.Txt | Fichier de macro-instructions pour 'fuseaux' |
| CalEqu.Txt | Fichier d'équivalences pour 'fuseaux' |

*Une suggestion pour le prochain numéro,*
*une question sur T_Pom's ou ad litteram,*
*un regret à formuler,*
*une hésitation sur un programme Pom's ?*

**emp11**

*Notre boîte à lettres sur CalvaCom est à votre disposition.*

# MicroSoft Works

## Essai

Philippe Mathieu

Les anciens utilisateurs de l'Apple // se souviennent sans doute d'un logiciel miracle (pour son époque) : AppleWorks, qui existe toujours et a été perfectionné depuis. Dès l'origine, ce premier vrai intégré pour micro-ordinateur proposait trois fonctions : tableur, gestion de fichier et traitement de texte. La communication entre les fonctions se faisait, au moins partiellement, par Copier/coller grâce à un Presse-papiers, certes rudimentaire, mais qui préfigurait celui du Macintosh. Les différents modules étaient gérés par des commandes homogènes, très conviviales (pour ce type de matériel sans souris) et aisément mises en œuvre. Une fonction d'aide était accessible en permanence.

Quand on aborde Microsoft Works, on est frappé par sa conception, rappelant sur bien des points celle d'AppleWorks, mais avec – évidemment – les progrès dus, d'une part au fonctionnement du Macintosh, d'autre part à la puissance du matériel.

Works propose, outre les trois outils précédents, un grapheur associé au tableur, un module de courrier personnalisé par liaison du traitement de texte et de la base de données, et un module très complet de télécommunications. Les liaisons entre modules utilisent systématiquement le Copier/Coller, cependant que les liaisons avec d'autres logiciels se font par *importation* ou *exportation* de documents, ou encore par l'intermédiaire du module de communication.

Pris individuellement, les outils sont classiques, possèdent les fonctionnalités indispensables, sans sophistication. Ils sont en conséquence faciles maîtriser et à mettre en œuvre.

Pris ensemble, on apprécie le passage instantané d'une fonction à une autre, par simple activation de la fenêtre voulue, et la liaison elle aussi presque instantanée par le Presse-papiers.

À noter une bogue gênante : Works, non seulement est incompatible avec l'Album, mais de plus une tentative d'utilisation se solde toujours par une "bombe" ! Espérons que Microsoft corrigera rapidement cette erreur.

## Le traitement de texte

Il ressemble beaucoup à MacWrite ou à Word. Il utilise classiquement une règle et possède les fonctions, également classiques, d'alignement, pagination, en-tête et bas de page, recherche et remplacement, etc.

### Ses "plus"

- la possibilité de juxtaposer face à face du texte et des dessins ou graphiques issus, soit de logiciels tiers, via le Presse-papiers, soit du grapheur intégré ;
- le transfert de caractéristiques de formatage d'un paragraphe à un autre par clic ;
- l'existence d'une panoplie d'outils semi-graphiques intégrés : traits, encadrements, ovales, avec diverses épaisseurs de traits.

### Ses "moins"

- l'absence de fonction de mise en colonnes du texte, qu'on peut cependant s'attendre à trouver dans un logiciel moderne ;
- l'absence de tabulateur décimal ;
- l'absence de glossaire et, quoi que l'importance soit sans doute moindre, de dictionnaire et de césure automatique.

## Le gestionnaire de fichiers

Il s'appelle un peu pompeusement Base de données, c'est en fait un honnête gestionnaire de fichiers, dont la conception ressemble beaucoup à celle de File (également proposé par Microsoft).

### Ses "plus"

- des possibilités très larges de sélection d'enregistrement par des associations de fonctions logiques ;
- de bonnes possibilités de caractérisation des champs et d'exécution de calculs sur les champs numériques ;
- des formatages des dates intéressants ;
- une grande facilité de définition d'un état, de sous-totaux et totaux, de sauts de lignes et de pages ;
- la possibilité de conserver des "gabarits" d'états.

### Ses "moins"

- pas de vue préalable des états à l'écran (seulement les résultats des totalisations) ;
- la limitation de la longueur d'un champ (nom + contenu) à une ligne de l'écran.

Par ailleurs, il ne faut pas perdre de vue que ce gestionnaire est mono-fichier.

## Le courrier personnalisé

En associant un texte avec un fichier (ou plusieurs), on peut créer aisément un système de courrier personnalisé travaillant sur tout le (ou sur une sélection du) fichier. Il suffit d'indiquer, en position, les rubriques des fichiers à utiliser lors de l'impression, par choix dans une liste de rubriques présentées à l'écran.

### Ses "plus"

• une facilité extraordinaire d'utilisation ;
• la possibilité de voir à l'écran ce que sera exactement tel ou tel document imprimé (l'enregistrement où se trouve le curseur étant utilisé pour cette vue préalable).

### Ses "moins"

• on ose à peine le dire : le fait qu'il faille redemander l'option de préparation pour chaque champ à intégrer dans le document de traitement de texte.

## Le tableur

Sa parenté avec le célèbre Multiplan, et peut-être plus encore avec Excel (deux logiciels signés Microsoft) est évidente. Sa capacité est de 230 colonnes et de 9999 lignes, ce qui est très confortable. On y trouve une large panoplie de fonctions, un travail en références absolues ou relatives, internes ou externes, des options de formatage variées, etc.

### Ses "plus"

• la transposition aisée des lignes et colonnes, notamment pour la construction de graphes ;
• de nombreuses fonctions mathématiques, logiques, statistiques et financières ;
• une bonne variété d'options de *collage spécial* ;
• la possibilté de découper la fenêtre en deux "panneaux", horizontalement ou verticalement ;
• la déplacement de cellules obtenu par clic.

### Ses "moins"

• des fonctions de date et de chaînes de caractères inexistantes ;
• pas de possibilité de créer des formats personnalisés (comme c'est la cas avec Excel) ;

Par ailleurs, il n'existe pas de macro-commandes associées au tableur.

## Le grapheur

Il est associé au tableur, et permet de tracer des courbes, des diagrammes à barres (éventuellement combinés) et des diagrammes à secteurs (camemberts). Un graphique est, comme il se doit, mis à jour automatiquement par la feuille de calcul qui le sous-tend.

### Ses "plus"

• la possibilité de représenter des lignes de données disjointes (ce que ne permet pas un outil aussi puissant qu'Excel !) ;
• une échelle semi-logarithmique disponible ;
• la possibilité de conserver des "gabarits" utilisables en diverses circonstances.

### Ses "moins"

• la limitation à 4 séries de valeurs ;
• la représentation de séries de lignes uniquement (pour représenter des séries de colonnes, il faut passer par l'intermédiaire de la fonction *Transpose*, heureusement aisée à mettre en œuvre).

## La fonction communication

Il est relativement rare de trouver aujourd'hui encore une fonction communication intégrée à un logiciel. Celle de Works est à la fois puissante et très commode




d'emploi. On peut définir sur un écran les caractéristiques techniques de la communication, puis envoyer ou recevoir des messages (frappe directe au clavier), des textes (sans formatage), des fichiers (avec toutes les caractéristiques de formatage, en texte ou en dessin ou graphique).

La rareté des options communication fait qu'il est difficile de parler de "plus" et de "moins" : par rapport à quoi ?

### Ce qu'on aime

• la très grande variété possible des spécifications techniques ;
• la possibilité de créer des "documents communications" qui mettent en mémoire les spécifications et un "annuaire téléphonique" personnalisé ;
• sous réserve de posséder le modem voulu, la numérotation automatique, la mise en réception automatique ;
• la facilité opératoire.

### Ce qu'on n'aime pas

• rien de particulier.

## La liaison entre outils

Elle utilise systématiquement le Copier/Coller, quels que soit les outils entre lesquels on veut établir un transfert de données, sauf:

• le courrier personnalisé, réalisé par une fonction particulière ;
• la liaison tableur-grapheur, qui est automatique.

### Ses "plus"

• l'instantanéité du passage d'un outil à un autre par ouverture (si

ce n'est déjà fait) ou activation de fenêtres ;
• la convivialité très poussée, à la Macintosh, du passage par le Presse-papiers.

## La liaison avec d'autres logiciels

Elle peut se faire de trois façons :

• par le truchement du Presse-papiers, comme entre les fonctions de Works ;
• par lecture ou enregistrement sous forme de texte seul, pour les logiciels qui comportent cette possibilité (mais les options de formatage sont perdues), ou sous certains formats particuliers, par exemple SYLK ;
• en utilisant le module de communication (pour les logiciels, Macintosh ou ordinateur tiers, qui disposent eux-mêmes d'un tel module ou peuvent en utiliser un).

### Ses "plus"

• de nombreuses solutions possibles, dans lesquelles on peut faire un choix en fonction du problème à traiter ;
• la grande facilité d'usage des liaisons par Presse-papiers ou module de communication.

### Ses "moins"

• la nécessité, si on sort des options simples, de bien comprendre la structure des documents produits par d'autres logiciels : format, caractères de contrôle,... (mais ce n'est pas propre à Works) ;
• surtout, c'est là que se fait durement ressentir l'incompatibilité de Works avec l'Album.

## En résumé

### Pour

Un logiciel pour lequel on est parfois tenté d'utiliser l'adjectif génial, en particulier pour son module de communication, sa fonction de courrier personnalisé et la facilité de commutation et de liaison entre outils.

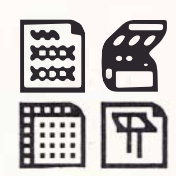
Une très grande facilité d'utilisation, parfois au détriment de la puissance, mais jamais au détriment de la rapidité.

### Contre

Quelques lacunes regrettables, comme l'absence de colonnage dans le traitement de texte ou de fonction de date dans le tableur. Et surtout, l'incompatibilité avec l'Album, générant une bombe et une 'grave erreur système'.

### Un bilan très largement positif

Works peut fort bien être le **logiciel unique** pour les besoins courant de la plupart de ces "travailleurs du savoir" chers à nos amis d'Apple, certains d'entre eux étant amenés par ailleurs à utiliser des logiciels spécialisés de leur profession ou de leur technicité.



# MOTS CROISES




Roland Jost

Une revue telle Pom's pouvait-elle proposer des grilles de mots croisés sans proposer l'assistance de l'ordinateur ? L'application présentée ici permet de traiter sur l'écran des grilles de mots croisés préenregistrées ; elle présente le double avantage de remplacer avantageusement crayon et gomme, et d'aborder un nouveau langage dans ces pages : le Turbo-Pascal. L'application permet de sauvegarder une recherche en cours pour la reprendre à tête reposée.

## Utilisation

Par le double-clic habituel, lancer l'application Mots_Croisés.

Par défaut, la grille numéro 1 est chargée. Si cette grille n'est pas présente sur la disquette, un message d'erreur apparaît. Pour sélectionner une autre grille (s'il en existe une), cliquer sur le rectangle `grille numéro`. Une boîte de sélection apparaît alors. Choisir le numéro de la grille en cliquant sur les cases + ou –, puis charger la grille en cliquant dans `OK`.

Déplacer la souris sur la grille, frapper la lettre à inscrire dans la case pointée par la souris ; bien sûr les cases noires ne sont pas accessibles.

Pour conserver une grille inachevée, cliquer `Sauver`. Cette grille pourra être rappelée dans l'état ultérieurement.

La commande `Effacer` vide la grille.

Il est possible d'imprimer le problème sur papier en cliquant… `Imprimer`.

Pour quitter le programme choisir l'option… `Quitter`.

Enfin, pour contrôler les résultats, enfoncer la touche `TAB` tout en cliquant dans `Grille numéro`. La solution du problème s'affiche (*Le problème de ce numéro faisant l'objet d'un concours, cette option ne fonctionnera pas…*).

## Structure d'un fichier Grille

Les fichiers grilles sont des fichiers TEXT nommés GRILLE1, GRILLE2.... Leur contenu doit être le suivant :

- une chaîne de 100 caractères où sera stockée la grille en cours de remplissage ;
- 1 retour chariot ;
- 1 chaîne de caractères contenant la solution du problème. Les cases noires sont représentées par des points ;
- 1 retour chariot ;
- 1 certain nombre de caractères pour les définitions horizontales ;
- 1 retour chariot ;
- 1 astérisque ;
- 1 retour chariot ;
- les caractères nécessaires pour les définitions verticales.

L'encadré donne un exemple de fichier.

## *Exemple de fichier exploitable par Mots_croises*

Pour plus de compréhension, les espaces ont été matérialisés par des •. Un tel fichier peut-être constitué à l'aide d'un traitement de textes (sauvegarde avec l'option 'texte seul').

```
•••••••••••••.••.•••••..••.•••••••••••.•••••..•.••••..•••.••••.••••.••..••••.•••••••••••••••••••••.•
INTERPRETENOE.OE.CUDTU..SU.UBUEVITERAS.CGERS..M.MAEA..LSE.OTRU.LION.YE..NAAN.LEUORDINATEURRESTERAI.S

1. Parle plusieurs langues. - 2.  Patriarche. - Cardi-
naux. - Noble inversé. - 3. Caché. - Appris. - Père de
 comédie. - 4. Feras un détour. - 5. Rivière de France. - Possessif. - 6. Voyelles. - Un
langage français qui a eu bien peu de succès.  Centre de vote. - 7.  Petit  ruisseau. - Le roi des a
nimaux.  Début de monnaie. - 8. - Quatre de Nathan.  Vieux loup.  - 9. De plus en  plus puissant. -
10. Ne partirai pas.
*
1.  Type de variable. - Opérateur logique. - 2. Récent. - Note. - 3. Inutile avec Mac Draw. - Fin de
 verbe
- Inversé :  Avait son siège à Genève. - 4. Consonnes . - Bon pour le nourrisson. - 5. Emblème d'un
parti
politique. - Grimpe aux arbres. - 6. Crainte. - Système de détection. - 7. Ainsi soit-il. - Symbole
chimique. - 8. Monnaies européennes. - Monnaie d'Europe. - 9. Récipient. - Axe. - 10. Enseignants.
```



# Programme Mots_Croises
## Turbo-Pascal

```
PROGRAM Mots_Croises;

{Roland JOST - VERSION 2.0 - Juin 1987}

{$U-}
{$T APPLMC02}

uses Memtypes, Quickdraw, OSIntf, ToolIntf, PackIntf,PasInOut,MacPrint;

const
     System= 0;
     NewYork = 2;

type
      str15 = string[15];
      CharArray = packed array[0..4999] of Char;

var
     i,
     j,
     ncase,
     fasc,
     fdesc,
     lin,
     nombre1,
     enmo              : integer;
     textinlength,
     textin1length     : longint;
     inRect,
     r,
     r1,
     arect,
     brect             : Rect;
     inkey             : char;
     encours,
     solution          : string[100];
     chaine,
     numero,
     s,
     nomgrille,
     sn                : string;
     c                 : point;
     cases             : array[1..100] of rect;
     commande          : array[1..5] of rect;
     mes               : array[1..4] of string[10];
     Line              : array[1..50] of string[100];
     gPort             : GrafPort;
     myEvent           : eventRecord;
     sortie,
     solflag           : boolean;
     Textin,
     Textin1,
     TextOut           : CharArray;
     Info              : FontInfo;

{=====================================================================}

procedure PutMsg (m : str15;
     r : Rect);

{ ecrit la chaine de caractères m dans le rectangle r }

var
 h, v : integer;
begin
 h := r.left + (r.right - r.left - StringWidth(m)) div 2;
 v := r.top + (r.bottom - r.top - fAsc - fDesc) div 2 + fAsc;
MoveTo(h, v);
 DrawString(m);
end;{PutMsg}

{=====================================================================}

procedure boite5;

{ affiche le numéro de la grille en cours }

begin
    SetRect(commande[5],150,10,230,30);
   FrameRoundRect(commande[5],5,5);
   PutMsg('grille n° ' + sn,commande[5]);
end; {boite5}

{=====================================================================}

procedure messages;

{ initialise et affiche les commandes }

begin

   SetRect(commande[1],10,10,70,30);
   FrameRoundRect(commande[1],5,5);
    PutMsg('Effacer',commande[1]);
    SetRect(commande[2],80,10,140,30);
   FrameRoundRect(commande[2],5,5);
   PutMsg('Sauver',commande[2]);
    SetRect(commande[3],10,40,70,60);
   FrameRoundRect(commande[3],5,5);
   PutMsg('Imprimer',commande[3]);
    SetRect(commande[4],80,40,140,60);
   FrameRoundRect(commande[4],5,5);
    PutMsg('Quitter',commande[4]);
   boite5;

end; {messages}
{=====================================================================}

procedure remplit_grille ;

{ affiche le contenu de la grille }

var i : integer;

begin
   for i := 1 to 100 do begin
     eraseRect(cases[i]);
     case encours[i] of
       '.' : invertrect(cases[i]);
      otherwise
        putmsg(encours[i],cases[i]);
     end; {case}
   end; {for i}

end; {remplit_grille}
{=====================================================================}

procedure affiche_solution ;

{ affiche la solution de la grille }

var i : integer;

begin
   for i := 1 to 100 do begin
     eraseRect(cases[i]);
     case solution[i] of
       '.' : invertrect(cases[i]);
      otherwise
        putmsg(solution[i],cases[i]);
     end; {case}
   end; {for i}
   solflag :=false;
end; {affiche_solution}

{=====================================================================}

procedure affiche_grille;

{ génère la grille 10 * 10 de mots croisés }

var   i, j : integer;

begin
```

```
     for j := 1 to 10 do
      begin
      for i := 1 to 10 do begin
           SetRect(r,270+20*i,20*j,290+20*i,20+20*j);
        FrameRect(r);
           SetRect(r1,271+20*i,1+20*j,289+20*i,19+20*j);
        cases[i+10*(j-1)] := r1;
     end;
          SetRect(r1,270,20*j,290,20+20*j);
        NumToString(j,chaine);
        PutMsg(chaine,r1);
     end;
      for i := 1 to 10 do begin
         SetRect(r1,270 + 20*i,10,290+20*i,20);
        Numtostring(i,chaine);
        PutMsg(chaine,r1);
     end;

 end; {affiche_grille}
 {===========================================================}

 Procedure faittextrect;

 var
    axrect, bxrect : rect;

 begin
    SetRect(axrect,5,80,270,210);
    SetRect(bxrect,5,230,507,330);

   aRect.left := axRect.left +5;
   aRect.right := axRect.right - 5;
   aRect.bottom := axRect.bottom - 5;
   aRect.top := axRect.top + 5;

   bRect.left := bxRect.left +5;
   bRect.right := bxRect.right - 5;
   bRect.bottom := bxRect.botlom - 5;
   bRect.top := bxRect.top + 5;

 end; { Procedure faittextrect }

{===========================================================}

procedure rappel;

{ chargement d'un problème à partir du disque }
{ et affichage des définitions. }

 var

  i,erreur : Integer;
  F : Text;

  car : char;

begin

  nomgrille := 'grille'+sn;
   {$I-}
  Reset(F, nomgrille);
  erreur := IOResult;
  Readln (F,encours);
  erreur := IOResult;
  Readln(F,solution);
  erreur := IOResult;
 if erreur = 0 then begin
  lin:=0;
  while car <> '*' do
   begin
    Read(F,car);
    Textin[lin] := car;
   lin:=lin + 1;
  end;
  textinlength := lin - 1;
  Read(F,car);
  lin := 0;
  while not Eof(F) do
  begin
   Read(F,Textin1 [lin]);
  lin :=lin + 1;
 end;
 textin1length := lin;
   end; {if}
  close(f);
   if erreur <> 0 then begin
    SysBeep(1);
    Moveto(200,150);
     DrawString('Cette grille n''existe pas !!');
   end
   else begin
    affiche_grille;
    remplit_grille;
     moveto(5,80); DrawString('Horizontalement');
      TextBox(@Textin,textinlength, arect, 0);
     moveto(5,225); DrawString('Verticalement');
      TextBox(@Textin1,textin1length, brect, 0);
   end;
   {$I+}

   end;{rappel}

{===========================================================}

procedure saisie_numero;

{ saisie du numéro de la grille à charger }

var    boite,              { contour }
       rc1,                { rectangle pour le - }
       rc2,                { rectangle pour le + }
       rok                 { rectangle pour OK }
           :rect;

 begin
   EraseRect(commande[5]);
    SetRect(boite,150,10,250,60);
   FrameRoundRect(boite,5,5);
    Moveto(170,25);DrawString('grille n° :');
   PenSize(1,1);
    SetRect(r1,220,14,240,29);
    SetRect(rc1,224,40,236,51);
    SetRect(rc2,170,40,182,51);
    PutMsg('+',rc1);
    PutMsg('-',rc2);
    SetRect(rok,190,35,214,55);
   FrameroundRect(rok,8,8);
    PutMsg('OK',rok);
    Frameroundrect(rc1,5,5);
   FrameRoundRect(rc2,5,5);
   case ord(sn[0]) of
    1 : nombre1 := ord(sn[1])-48;
     2 : nombre1 := 10*(ord(sn[1])-48) +ord(sn[2])-48;
     3 : nombre1 := 100*(ord(sn[1])-48) + 10*(ord(sn[2])-48) +ord(sn[3])-48;
  end; {case}
   PutMsg(sn+'.',r1 );
   repeat
    Getmouse(c);
     if Button and PtInRect(c,rc1) then begin
    nombre1 := nombre1+1;
    if nombre1 > 999then nombre1 := 1;
     NumToString(nombre1,sn);
     EraseRoundRect(r1,5,5);
      PutMsg(sn+'.',r1);
      for j := 1 to 30000 do;
     end;{if}
      if Button and PtInRect(c,rc2) then begin
    nombre1 := nombre1-1;
    if nombre1 < 1 then nombre1 := 1;
     NumToString(nombre1,sn);
     EraseRect(r1);
      PutMsg(sn+'.',r1);
      for j := 1 to30000do;
     end;{if}
    until (button and PtInRect(c,rok));

   c.h := 10;
   c.v :=10;
    EraseRect(inRect);

   messages;
   rappel;
end; {saisie_numero}

{===========================================================}

procedure sauve_grille;
```

```
{ sauvegarde de la grille en cours sur disque }

var i : integer;
   f : text;

begin
  rewrite(f, 'grille'+sn);
  writeln(f,encours);
  writeln(f,solution);
  for i := 0 to textinlength -1 do
     write(f,textin[i]);
  writeln(f,'*');
  for i := 0 to textin1length - 1 do
     write(f,textin1[i]);
  close(f);
end; {sauve_grille}

{=========================================================}
procedure quelle_case;

{ trouve la case pointée par le curseur }

var i : integer;
   c : point;

begin

  for i := 1 to 100 do
  begin
  Getmouse(c);
   if PtInrect(c,cases[i]) then ncase := i;
  end;

end; {quelle_case}

{=========================================================}

procedure saisie(myEvent : EventRecord);

{ filtre les touches clavier et affiche dans la grille }

begin

  quelle_case;
  if encours[ncase] <> '.' then
  begin
  If (myevent.Message and charcodemask) > 96 then myevent.message := myevent.message - 32;
  inkey := char(myevent.message and charcodemask);
    case inkey of
     'A'..'Z' : begin
             eraseRect(cases[ncase]);
             PutMsg(inkey,cases[ncase]);
             Delete(encours,ncase,1);
             Insert(inkey,encours,ncase);
             SysBeep(-1);
            end;
     otherwise
        sysbeep(1);
    end; {case}
   end {if}
  alse sysbeep(2);
end; {saisie}

{=========================================================}

procedure efface;

{ efface le contenu de la grille }

var i : integer;
   f : text;

begin

 for i := 1 to 100 do
 if (encours[i] in ['A'..'Z','a'..'z']) then encours[i] := ' ';

end; {efface}


{=========================================================}

procedure bouton;

{ analyse des commandes et branchement vers les routines concernées }

var c    : point;
   delai : longint;

begin
  GetMouse(c);
  if PtInRect(c,commande[1]) then begin
   efface;
    remplit_grille;
   sauve_grille;
   end; {if}
  if PtInRect(c,commande[2]) then sauve_grille;
  if PtInRect(c,commande[3]) then begin
   HideCursor;
   PrOpen;
      PrCtlCall(iPrEvtCtl,lPrEvtAll,0,0);
   PrClose;
   ShowCursor;
   end; {if}
  if PtInRect(c,commande[4]) then sortie := true;
  if PtInRect(c,commande[5]) then begin
    if solflag = true than
    affiche_solution
   else
    saisie_numero;
   end; {if}
  c.h := 10;
  c.v:=10;
end; {bouton}
```

*Votre premier problème est page 73...*

```
{=========================================================}

procedure Initialise;

{ initialisation }

         begin
              initgraf(@thePort);
             openport(@gport);
            PenPat(black);
            BackPat(white);
             FrameRect(gPort.portRect);
            inRect:= gPort.portRect;
            InsetRect(inRect,1,1);
            EraseRect(inRect);
            TextFont(NewYork);
            TextSize(9);
            GetFontInfo(Info);
            initcursor;
            sortie := false;
            solflag :=faise;
            encours:= '';
            with Info do
            begin
             fasc := ascent;
             fdesc := Descent;
            end;
             tailtextrect;
            sn := '1';
            rappel;
            messages;

         end; {initialise}

{=========================================================}

begin {main}

 initialisa;

  repeat
   if GetNextEvent(everyEvent, myEvent) then
   case myEvent.what of
   MouseDown : bouton;
   KeyDown : begin
          If (myevent.message and charCodeMask) = 9 then solflag := true;
          Saisie(myEvent);
         end;
   end;{case}
 until SORTIE;


end.
```




# Alain Bohec

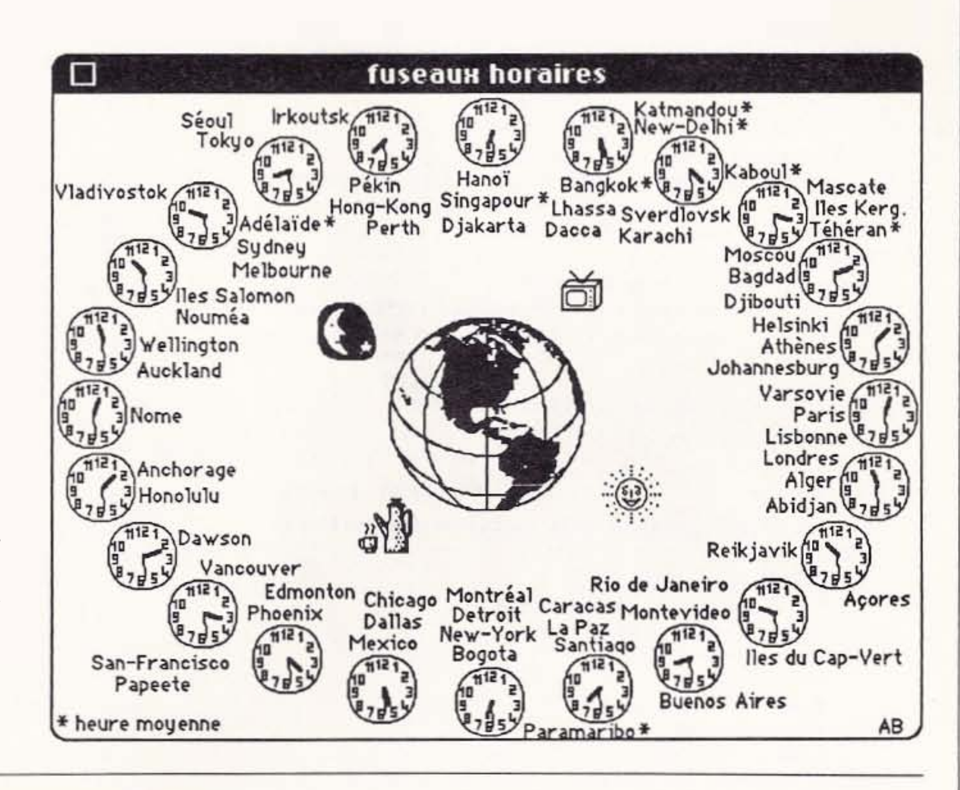
# Fuseaux horaires

Connaître instantanément l'heure de Vladivostok, Phoenix ou — accessoirement — Paris... Tel est le but de cet accessoire de bureau écrit en assembleur 68000 pour l'efficacité mais aussi, inutile d'essayer de le dissimuler, pour le plaisir.

Écrire un mode d'emploi de *fuseaux* tiendrait du gag tant son usage est 'transparent' ; signalons seulement que cafetière, tasse, lune, téléviseur et autre soleil indiquent ce que sont censés faire à cette heure — et en gros — vos amis de Vancouver ou Singapour.

L'installation de *fuseaux* dans le 'system' de vos disquettes de démarrage est aussi simple que pour d'autres accessoires et *Font/DA Mover* fera très bien l'affaire. Si vous êtes nouveau lecteur de Pom's (Bienvenue !), il convient de signaler que, comme d'habitude, la disquette d'accompagnement de ce numéro contient l'accessoire installé dans le système et dans un fichier Font/DA Mover. Les 'sources' sont aussi sur ladite disquette Mac 32.

Les virtuoses de l'éditeur de ressources vont aussi pouvoir s'occuper : les noms de ville ne sont pas dans le code mais dans une ressource (DITL -16000 pour ne rien vous cacher), il est donc possible de remplacer, par exemple, Paris par Brest ou Bordeaux.




## Source 'Fuseaux.Asm'

```
*                    • FUSEAUX •
*              heure à travers le monde
*        A2 : handle de la liste des points
*             DialogPtr
*        A3 : adresse du BitMap du cadran
*             adresse du handle d'un objet tournant
*        A4 : sauvegarde du DCEPtr
*        D3 : compteurs
*        D4 : pivot
* PRELIMINAIRES ...................................
* INCLUDES ----------------------------------------
include QuickEqu.D
include SystEqu.D
include ToolBEqu.D
include MacTraps.D
include CalEqu.D
include mABcro.Txt
* EQUATES -----------------------------------------
csCode   equ     $1A      ; control/status code [W]
csParam  equ     $1C      ; op-defined params [20*B]
NHeur    equ     12
NFus     equ     24
* écarts dans la liste des Points ------------------
Piv      equ     0        ;pivot
LAig     equ     4        ;longueurs des aiguilles
Ax       equ     8        ;axe de rotation
Ell      equ     12       ;petit et grand axe ellipse
* X -----------------------------------------------
* xRef --------------------------------------------
         xRef    GrandAig
         xRef    PetitAig
```

```
        xRef    GetTime
        xRef    SinCos
        xRef    Rotvect90
* xRef -------------------------------------------------
                xDef    dt
                xDef    DERU
                DERU :
* MOTS D'ENTREE ----------------------------------------
* Comportement -----------------------------------------
        dc.W    $2400   ;ctrl/actions périodiques
        dc.W    3600    ;toutes les minutes
        dc.W    $0040   ;uniquement èvents mise à jour
        dc.W    0       ;pas de menu
* Ecarts vers les Routines -----------------------------
        dc.W    DIGORADYR-DERU    ; 1:ouverture
        dc.W    HorolazhEshy-DERU ; prime (unused)
        dc.W    YRZ-DERU          ; 2:controle
        dc.W    HorolazhEshy-DERU ; status (unused)
        dc.W    KLOZADYR-DERU     ; 3:fermeture
* Titre ------------------------------------------------
talbenn dc.B    0
        dc.B    ''
        .ALIGN  2
* OUVERTURE @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
DIGORADYR
* PRELIMINAIRES ....................................
                emp M.L,A2/A4   ;sauvegarde
        moveA.L A1,A4           ;DCEPtr » registre sûr
* Sauvegarde du GrafPtr de l'application en cours ----
        ep L            ;pour le GrafPtr
        emp .L,SP       ;GrafPtr » pile
        _GetPort        ;GrafPtr APPL en cours » pile
        _ObscureCursor  ;pour faire plus joli
* QUESTION DE LA FENETRE ++++++++++++++++++++++++++++++
        tst.L   dctlWindow(A4)  ;déja fenêtre ?
                bne.W   Prenestrameus
* CONSTRUCTIONS ....................................
; si l'on n'a pas de fenêtre, tout est à construire
* COLLECTE DE L'ID DE BASE ++++++++++++++++++++++++++++
        move    dCtlRefNum(A4),D0       ;n° de l'Acc
        addQ    #1,D0
        neg     D0                      ;sous-ID Acc
        lsl     #5,D0
        move    #$C000,D1
        or      D0,D1
        emp .W,D1       ;ID de base surpile et dans D1
* LISTE DE POINTS +++++++++++++++++++++++++++++++++++++
* Collecte de la Liste et Placement ------------------
        GetRes #'pnt#',D1
        dep A.L,A2      ;handle dans A2
        moveA.L (A2),A0 ;adresse
        lea     Pivot,A1
        move.L  Piv(A0),(A1)+
        move.L  Laig(A0),(A1)+
        move.L  Ax(A0),(A1)+
        move.L  Ell(A0),(A1)    ;axes de l'ellipse
* Destruction ----------------------------------------
        emp .L,A2
        _ReleaseResource
; Si l'on n'a pas déja une fenêtre, +++++++++++++++++
* il faut en fabriquer une :
* Placement de l'ID de la Mappemonde ----------------
* dans l'Item No 25
        GetRes #'DITL',D1
        dep A.L,A0      ;handle
        moveA.L (A0),A0
        move D1,Wo+NFus*(itmData+By+Wo)+itmData+By(A0)
* Collecte du Dialogue ------------------------------
* pointeur ++++++++++++++++++++++++++++++++++++++++++++
        ep L                    ;pour le DialogPtr
        emp .W,D1
        emp .L,#0               ; » heap
        emp .L,#-1              ;fenêtre premier plan
        _GetNewDialog           ;fenêtre fabriquée !
        moveA.L (SP),A2         ;DialogPtr » A2 & pile
* fonte ++++++++++++++++++++++++++++++++++++++++++++++++
        _SetPort        ;enlèvement du DialogPtr
        SetFont #geneva,#9
* finitions +++++++++++++++++++++++++++++++++++++++++++
        move.L  A2,dctlWindow(A4)       ;WindowPtr>DCE
        move    dctlRefNum(A4),WindowKind(A2)
* COLLECTE DES IMAGES +++++++++++++++++++++++++++++++++
* Cadran -----------------------------------------------
                ;ID principal toujours sur la pile
                dep.W,D1        ;ID » D1, RAN
        GetRes #'bmap',D1
        mouv0 L,(SP)+,CadrHndl
* Images Tournantes ------------------------------------
        lea     ObjTour,A1      ;adresse de stockage
        moveQ   #4,D2           ;compteur
@10     addQ    #1,D1
        ep L
        emp .W,D1
        _GetPicture
        dep .L,(A1)
        subQ    #1,D2
        beq.S   @20
        addA    #2*Lo,A1
        bra.S   @10
* POSITION INITIALE DE LA LUNE ++++++++++++++++++++++++
@20
* Collecte de l'Horloge Marquant 1h du Matin ---------
* c'est la 23ème, à une congruence prés,
* aprés l'horloge locale
        bsr.W   GetTime
        clr.L   D0
        move    dt+dtHour,D0
        add     #NFus-1,D0
        divU    #NFus,D0
        swap    D0      ;n° de l'item de 2h du mat
* TopLeft du rectangle ++++++++++++++++++++++++++++++++
        subA    #2*Extd,SP ;Rect/Hndl/type, étendus
        bsr.W   GetRect
        move.L  Axe,D0
        sub.L   D0,TopLeft(SP)
        OpBin Top(SP),VectLun+vv,I2X
        OpBin Left(SP),VectLun+hh,I2X
* Calculs ----------------------------------------------
* somme des carrès & norme ++++++++++++++++++++++++++++
        OpBin VectLun+vv,vv+Lo(SP),X2X
        OpBin VectLun+hh,hh+Lo(SP),X2X
        OpBin vv(SP),vv+Lo(SP),mulX
        OpBin hh(SP),hh+Lo(SP),mulX
        OpBin hh(SP),vv+Lo(SP),addX
        pea     (SP)
        emp .W,#foSQRT
        _FP68K
* vecteur unitaire ++++++++++++++++++++++++++++++++++++
        pea     (SP)
        pea     VectLun+vv
        emp .W,#foDiv+ffExt
        _FP68K
        pea     (SP)
        pea     VectLun+hh
        emp .W,#foDiv+ffExt
        _FP68K
        addA    #2*Extd,SP      ;RAN
* vecteur exact +++++++++++++++++++++++++++++++++++++++
        OpBin VEll,VectLun+vv,mulI
        OpBin VEll,VectLun+hh,mulI
* CALCUL ...........................................
* pour avoir les aiguilles de suite
                bsr.W   CalAig
* ACHEVEMENT .......................................
Prenestrameus
Distro
        _SetPort        ;ancien GrafPtr sur la pile
        moveA.L A4,A1
                dep M.L,A2/A4   ;restauration
HorolazhEshy
```

```
        moveQ   #0,D0     ; return no error
                          ;Kenavo
                          RTS
* FERMETURE @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
KLOZADYR
* SAUVEGARDES ..........................................
                emp M.L,A2/A4     ;sauvegarde
        moveA.L A1,A4     ;DCEPtr
        ep L
        emp .L,SP
        _GetPort          ;GrafPtr en vigueur » pile
* LIQUIOATIONS .........................................
        emp .L,CadrHndl
        _ReleaseResource
        lea      ObjTour,A1
        moveQ    #4,D1    ;compteur
@0      emp .L,(A1)
        _ReleaseResource
        subQ     #1,D1
                 beq.S    @10
        addA     #2*Lo,A1
                 bra.S    @0
@10     emp .L,dctlWindow(A4)    ;/pile pour destruc.
        clr.L    dctlWindow(A4)  ;sinon, boum !
        _DisposDialog            ;plus besoin fenêtre
                 bra.S    Distro ;on va fermer
* CONTROLE @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
YRZ
* ROUTINE PRINCIPALE & BRANCHEMENTS ....................
* A0 pointe sur le "Device Request Block" et A1
* sur le DCE
        emp .L,A4         ;sauvegarde
        moveA.L A1,A4     ;DCEPtr
        move     csCode(A0),D0            ;n° requête
        SSi I.W,#accEvent,D0,doCtlEvent ;événement
        SSi I.W,#accRun,D0,ActMin       ;action périod
YrzEshy
        moveA.L A4,A1     ;DCEPtr, pour jIODone
        dep A.L,A4        ;restauration
        moveQ   #0,D0     ; return no error
        emp .L,jIODone    ;"request" traité, saut IODone
                          RTS
* EVENEMENT ............................................
doCtlEvent
* ROUTINE PRINCIPALE & BRANCHEMENTS ....................
                emp .L,A2         ;sauvegarde
        moveA.L csParam(A0),A0    ;sur l'événement
        moveA.L EvtMessage(A0),A2 ;WindowPtr
        move     EvtNum(A0),D0    ;numéro évènement
        SSI I.W,#updatEvt,D0,MAJ  ;mise à jour
CtlEvtDone
                dep A.L,A2        ;restauration
                        bra.S     YrzEshy
MAJ
* MISE A JOUR ..........................................
*entrée : DialogPtr » A2
        emp .L,A2
        _BeginUpdate
        emp .L,A2
        _DrawDialog
                bsr.W    DessAutour
        emp .L,A2
        _EndUpdate
                bra.S    CtlEvtDone
* ACTION PERIODIQUE ....................................
ActMin
* MECANISME ............................................
                bsr.S    CalAig
                bsr.W    DessAutour
                bra.S    YrzEshy
* SOUS-ROUTINES @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
CalAig
* CALCUL DES POSITIONS .................................
* PREPARATIFS ++++++++++++++++++++++++++++++++++++++++++++
                bsr.W    GetTime
```

```
        moveA.L dctlWindow(A4),A2
                subA     #3*Extd,SP ;angle/vecteur
*  POSITION DE BASE ++++++++++++++++++++++++++++++++++++++
* Grande Aiguille ---------------------------------------
        moveA.L SP,A0
        lea      ga,A1    ;adresse du résultat
        move     Lga,D0   ;longueur
                bsr.W    GrandAig
* Petite Aiguille ---------------------------------------
        moveA.L SP,A0
        lea      pa,A1    ;adresse du résultat
        move     Lpa,D0   ;longueur
                bsr.W    PetitAig
* petite aiguille en étendu ++++++++++++++++++++++++++++
* multiplication du vecteur unitaire orthogonal
* par la longueur
* calcul ------------------------------------------------
        pea      Lpa
        pea      Extd+vv+Lo(SP)
        fMulI
        pea      Lpa
        pea      Extd+hh+Lo(SP)
        fMulI
        ;reste pa en étendu sur la pile + 1 espace
* rotation de 90 ---------------------------------------
;dessous » dessus
        pea      Extd+hh(SP)
        pea      Lo(SP)
        fX2X
;milieu » dessous
        pea      Extd+vv(SP)
        pea      Extd+hh+Lo(SP)
        fX2X
;dessus » milieu
        pea      (SP)
        pea      Extd+vv+Lo(SP)
        fX2X
;remplacement par l'orthogonal
        move     #-1,(SP)
        pea      (SP)
        pea      Extd+vv+Lo(SP)
        fMulI
* AUTRES POSITIONS ++++++++++++++++++++++++++++++++++++++
*       on fait tourner le petite aiguille de pi/6
* Initialisations ---------------------------------------
        lea      Extd(SP),A1 ;adr vecteur à tourner
                pea      pa  ;place résults int » pile
        move     #NHeur-1,D0 ;compteur
* Boucle ------------------------------------------------
@0      lea      Sin5,A0 ;adr lignes trigo de alfa
                bsr.W    RotAlfa
* transferts +++++++++++++++++++++++++++++++++++++++++++
        moveA.L (SP),A0
        addA     #Lo,A0
        move.L   A0,(SP)
        OpBin Extd+vv+Lo(SP),v(A0),X2I  ;r.sin(d+x)
        OpBin Extd+hh+Lo(SP),h(A0),X2I  ;r.cos(d+x)
* fini ? +++++++++++++++++++++++++++++++++++++++++++++++
        subQ     #1,D0
                bne.W    @0
                addA     #3*Extd+Lo,SP    ;RAN
* LUNE & SOLEIL +++++++++++++++++++++++++++++++++++++++++
* Coordonnées Relatives de la Lune ----------------------
        lea      SinMin,A0   ;adr lignes trigo de alfa
        lea      VectLun,A1  ;vecteur à faire tourner
                bsr.W    RotAlfa
        OpBin VectLun+vv,LunI+v,X2I
        OpBin VectLun+hh,LunI+h,X2I
* Calcul de toutes les Coordonnées Absolues -------------
        lea      LunI,A0
        moveQ    #4,D1    ;compteur
@10     move.L   (A0),D0
        move     D0,D2
        mulS     HEll,D2
        divS     VEll,D2
```

```
        move    D2,D0
        SWAP    D0
        add     Axe+v,D0
        SWAP    D0
        add     Axe+h,D0
        subO    #1,D1
                beq.S   @20     ;c'est fini
        move.L  (A0),2*Lo(A0)
        move.L  D0,(A0)
        addA    #2*Lo,A0
                bsr.W   RotVect90
                bra.S   @10
@20     move.L  D0,(A0)
                        RTS
DessAutour
* DESSIN ..........................................
* CADRANS & AIGUILLES +++++++++++++++++++++++++++++++
                emp M.L,A3-A4/D3-D4     ;sauvegarde
* Initialisations ---------------------------------
        emp .L,A2
        _SetPort
        moveA.L CadrHndl,A0
        moveA.L (A0),A3         ;adresse du bmap
        lea     BitMapRec(A3),A0 ;adr de la bitimage…
        move.L  A0,BaseAddr(A3)  ;… à la bonne place
        lea     pa,A4           ;adr dims aiguilles
        move    #NFus,D3        ;compteur
        move.L  Pivot,D4        ;pour la translation
        subA    #LR+Lo+Wo,SP    ;Rect/Hndl/type
* Boucle -------------------------------------------
@0
* collecte du rectangle ++++++++++++++++++++++++++++
        move    D3,D0
                bsr.W   GetRect
* cadran ++++++++++++++++++++++++++++++++++++++++++++
        move.L  TopLeft(SP),D0
        add.L   Bounds+BotRight(A3),D0
        sub.L   Bounds+TopLeft(A3),D0
        move.L  D0,BotRight(SP)
        emp .L,A3
        pea     Bounds(A3)      ;SrcRect=BitMapBounds
        pea     2*Lo(SP)        ;destRect
        emp .W,#srcCopy         ;écrase
        emp .L,#0               ;pas de ClipRgn
        _StdBits
* ai uilles ++++++++++++++++++++++++++++++++++++++++
        add.L   D4,(SP)
* grande aiguille ----------------------------------
        emp .L,(SP)
        _MoveTo
        emp .L,ga
        _Line
* petite aiguille ----------------------------------
        emp .L,#$20002
        _PenSize        ;plus large que grande aiguille
        emp .L,(SP)
        _MoveTo
        emp .L,(A4)
        _Line
        _PenNormal      ;retour aux dimensions normales
* Sortie ------------------------------------------
        subQ    #1,D3
                beq.S   @20     ;sortie
        cmpI.W  #NHeur,D3
                bne.S   @10     ;mystère des macros
        lea     pa,A4           ;réintialisation/pa
                bra.S   @0
@10     addA    #Lo,A4          ;cas ordinaire
                bra.S   @0
* LUNE & SOLEIL ++++++++++++++++++++++++++++++++++++
@20     lea     ObjTour,A3
        moveQ   #4,D3   ;compteur
@30     move.L  Lo(A3),TopLeft(SP)
        move.L  Lo(A3),BotRight(SP)
        moveA.L (A3),A0 ;handle
        moveA.L (A0),A1 ;adresse
        move.L  PicFrame+TopLeft(A1),D0
        move.L  PicFrame+BotRight(A1),D1
        sub.L   D0,D1
        add.L   D1,BotRight(SP)
        emp .L,A0
        pea     Lo(SP)
        _DrawPicture
        subQ    #1,D3
                beq.S   @40
        addA    #2*Lo,A3
                bra.S   @30
* FINITIONS ++++++++++++++++++++++++++++++++++++++++
@40             addA    #LR+Lo+Wo,SP    ;RAN
                dep M.L,A3-A4/D3-D4     ;restauration
                RTS
GetRect
* COLLECTE D'UN RECTANGLE D'ITEM ....................
*entrée : au moins 14 octets libres » pile
*         n° de l'item » D0
*sortie : rectangle au sommet de la pile
;PC sur la pile
        emp .L,A2
        emp .W,D0
        pea     Lo+Wo+Lo+LR+Lo(SP)
        pea     Lo+Lo+Wo+Lo+LR(SP)
        pea     Lo+Lo+Lo+Wo+Lo(SP)
        _GetDItem  ;TopLeft sommet de la pile #sous PC
        RTS
RotAlfa
* ROTATION D'UN VECTEUR DE L'ANGLE ALFA ..............
*entrée : adresse de SinCosAlfa » A0
*         adresse du vecteur à faire tourner » A1
*sortie : A0 » sans changement
*         A1 » adresse du vecteur transformé
                subA    #2*Extd,SP
        OpBin vv(A1),vv+Lo(SP),X2X
        OpBin hh(A1),hh+Lo(SP),X2X
        OpBin vv(A0),vv(A1),mulX        ;r.sind.sinx
        OpBin hh(A0),hh(A1),mulX        ;r.cosd.cosx
        OpBin hh(A0),vv+Lo(SP),mulX     ;r.cosd.sinx
        OpBin vv(A0),hh+Lo(SP),mulX     ;r.sind.cosx
        pea     hh(SP)
        pea     vv+Lo(SP)
        emp .W,#ffEXT+foADD
        _FP68K                  ;r.sin(d+x) » pile
        pea     vv(A1)
        pea     hh(A1)
        emp .W,#ffEXT+foSUB
        _FP68K                  ;r.cos(d+x) » bonne adr
        OpBin vv(SP),vv(A1),X2X ;sin(d+x) » bonne adr
                addA    #2*Extd,SP
                        RTS
* DONNEES @@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
* CONSTANTES .......................................
Sin5    dc.W    $3FFE,$8000,$0000,$0000,$0000
Cos5    dc.W    $3FFE,$DDB3,$D742,$C265,$539E
SinMin  dc.W    $3FF7,$8EFA,$1757,$14B7,$0DBF
CosMin  dc.W    $3FFE,$FFFF,$604B,$01C2,$6FBC
* RESERVATIONS DE MEMOIRE ..........................
* TEMPS ++++++++++++++++++++++++++++++++++++++++++++
dt      dcB.W   7,0     ;fiche de la date et du temps
* LISTE DE POINTS ++++++++++++++++++++++++++++++++++
Pivot   dc.L    0       ;coin du rectangle » pivot
Lga     dc.W    0       ;longueur grandes aiguilles
Lpa     dc.W    0       ;longueur petites aiguilles
Axe     dc.L    0
VEll    dc.W    0       ;petit axe de l'ellipse
HEll    dc.W    0       ;grand axe -"-
* ROTATION +++++++++++++++++++++++++++++++++++++++++
* Aiguilles & Cadran --------------------------------
CadrHndl        dc.L    0       ;BitMapHandle du cadran
ga      dc.L    0       ;grande aiguille
pa      dcB.L   NHeur,0 ;petites aiguilles
* Autres Objets ------------------------------------
```

```
VectLUn dcB.B   2*Extd,0   ;emplacement lune étendu
ObjTour ;ne pas changer l'ordre :
LunHndl         dc.L    0
LunI            dc.L    0  ;emplacement lune en entier
TassHndl        dc.L    0
TassI           dc.L    0  ;emplacement tasse entier
SolHndl         dc.L    0
SolI            dc.L    0  ;emplacement soleil entier
TelHndl         dc.L    0
TelI            dc.L    0  ;emplacement télé en entier
                               FIN
```

# Fichier 'mABcro.Txt'

```
*                         • mABcro •
*                         ••••••••••
*                       Macros Géniales
*••••••••••••••••••••••••••••••••••••••••••••••••••••••
yaducalcul      equ     0
* EQUATES ••••••••••••••••••••••••••••••••••••••••••••••
Zo      equ     0
By      equ     1
Wo      equ     2
Lo      equ     4
LR      equ     8       ;rectangle
Extd    equ     10      ;longueur d'un étendu
vv      equ     0
hh      equ     10      ;points en étendu
LStr    equ     256     ;phrase
* DIVERS •••••••••••••••••••••••••••••••••••••••••••••••
* _SetFont_ ——————————————————————————————————————————
        MACRO SetFont fonte,taille =
        MOVE.W  {fonte},-(SP)
        _TextFont
        MOVE.W  {taille},-(SP)
        _TextSize
        |
* _P2Couic_ ——————————————————————————————————————————
*Traite toutes les séquences dans lesquelles avant
*d'appeler une routine on pousse deux entiers sur
*la pile (MoveTo,OffSetRect, … )
*-----------------------------------------------------
        MACRO P2Couic x,y,couic =
        MOVE.W  {x},-(SP)
        MOVE.W  {y},-(SP)
        _{couic}
        |
* PETITS BOUTS ———————————————————————————————————————
        MACRO tuu =
        moveM.L A0-A1/D0-D2,-(SP)
        move.W  #1,-(SP)
        _SysBeep
        moveM.L (SP)+,A0-A1/D0-D2
        |
        MACRO FIN =
        END
        |
        MACRO eP L =
        CLR.{L} -(SP)
        |
        MACRO deP f,R =
        MOVE{f} (SP)+,{R}
        |
        MACRO emP f,R =
        MOVE{f} {R},-(SP)
        |
        MACRO mouv0 L,D,Addr =
        LEA             {Addr},A0
        MOVE.{L}        {D},(A0)
        |
        MACRO mouv1 L,D,Addr =
        LEA             {Addr},A1
        MOVE.{L}        {D},(A1)
        |
        MACRO Si f,x,y,a =
        CMP{f}  {x},{y}
        BEQ     {a}
        |
        MACRO SiNon f,x,y,a =
        CMP{f}  {x},{y}
        BNE     {a}
        |
        MACRO SSi f,x,y,a =
        CMP{f}  {x},{y}
        BEQ.S   {a}
        |
        MACRO SSiNon f,x,y,a =
        CMP{f}  {x},{y}
        BNE.S   {a}
        |
* RESSOURCES •••••••••••••••••••••••••••••••••••••••••••
* _GetRes_ ——————————————————————————————————————————
* Macro générale pour la collecte des ressources
*-----------------------------------------------------
* Paramètres : type,ID » handle sur la pile
*-----------------------------------------------------
        MACRO   GetRes TYPE,ID =
        CLR.L           -(SP)
        MOVE.L  {TYPE},-(SP)
        MOVE.W  {ID},-(SP)
        _GetResource
        |
* _CollRes_ ——————————————————————————————————————————
* Macro pour la collecte de nombreuses ressources
*-----------------------------------------------------
*Pour obtenir un handle sur une ressource, on fournit
*souvent un ID et une instruction telle que
*GetPicture retourne un Handle
*-----------------------------------------------------
        MACRO   CollRes ID,Genre =
        CLR.L           -(SP)
        MOVE.W  {ID},-(SP)
        _Get{Genre}
        |
* _CollPhrase_ ———————————————————————————————————————
*Register-stack-based-macro pour les
*ressources de type STR#
*-----------------------------------------------------
*Macro permettant la collecte des pointeurs sur les
*phrases d'une liste
*(STR#Handle dans A0,No dans la liste dans D0 qui
*sert aussi de compteur) :
*STR Ptr sur la pile. Utilise aussi D1.
*-----------------------------------------------------
        MACRO CollPhrase =      ;donne Ptr/phrase
        MOVE.W  #2,D1           ;longueur des phrases
        MOVEA.L (A0),A0         ;adresse » A0
@100    ADDA.W  D1,A0           ;ptr/phrase dans A0
        MOVE.B  (A0),D1         ;long phrase dans D1
        ADDQ.W  #1,D1           ;cause octet longueur
        SUBQ.W  #1,D0           ;décrément du compteur
                BNE.S   @100    ;arrivé  bonne phrase
        MOVE.L A0,-(SP)
        |
*••••••••••••••••••••••••••••••••••••••••••••••••••••••
        IF yaducalcul
* CALCUL •••••••••••••••••••••••••••••••••••••••••••••••
* macros les plus utilisées de SANE
* MACROS OFFICIELLES ——————————————————————————————————
        MACRO   fX2X =  ;étendu->étendu
        MOVE.W  #FFEXT+FOZ2X,-(SP)
        _FP68K
        |
        MACRO   fX2I =  ;étendu->entier
        MOVE.W  #FFINT+FOX2Z,-(SP)
        _FP68K
        |
        MACRO   fI2X =  ;entier->étendu
        MOVE.W  #FFINT+FOZ2X,-(SP)
```

```
        _FP68K
        |
        MACRO   fmulI = ;multiplication par un entier
        MOVE.W  #FFINT+FOMUL,-(SP)
        _FP68K
        |
        MACRO   fMulX =  ;Multiplication par un étendu
        move    #ffEXT+foMUL,-(SP)
        _FP68K
        |
        MACRO   faddX = ;addition d'un étendu
        MOVE.W  #FFEXT+FOADD,-(SP)
        _FP68K
        |
        MACRO   faddS = ;addition d'un single
        MOVE.W  #FFSGL+FOADD,-(SP)
        _FP68K
        |
        MACRO   fmulS = ;multiplication par un single
        MOVE.W  #FFSGL+FOMUL,-(SP)
        _FP68K
        |
        MACRO   fdivI = ;division par un entier
        MOVE.W  #FFINT+FODIV,-(SP)
        _FP68K
        |
        MACRO   fremI =                     ;entier
        MOVE.W  #FFINT+FOREM,-(SP)
        _FP68K
        |
        MACRO   fCOSx = ;cosinus
        MOVE.W  #FOCOSX,-(SP)
        _Elems68K
        |
        MACRO   fSINx = ;sinus
        MOVE.W  #FOSINX,-(SP)
        _Elems68K
        |
*22 PERSO ----------------------------------------------
* Opération binaire-------------------------------------
*De nombreuses opérations du SANE utilisent deux
*arguments et un code d'opération. La macro
*ci-dessous reproduit la forme générale
*de la suit d'instructions correspondante.
*-------------------------------------------------------
        MACRO OpBin A,B,op =
        PEA     {A}
        PEA     {B}
        F{op}
        |
*-------------------------------------------------------
                                ENDIF
*.......................................................
                                FIN
```

## Fichier 'CalEqu.Txt'

```
*               ...............
*               • CalEqu.Txt •
*               ...............
*-------------------------------------------------------
*          Les equates ci-dessous font partie du
*      "Standard Apple Numeric Environment" (SANE)
*          décrit dans l'"Apple Numerics Manual"
*             inclus dans "Inside Macintosh".
*-------------------------------------------------------
* extraits de SANEMacs.Txt le 24 Juin 87
* condensé ?
*.......................................................
*.......................................................
* NUMEROS DES OPERATIONS ------------------------------
foADD           equ     0       ;Addition
foSUB           equ     2       ;soustraction
foMUL           equ     4       ;Multiplication
foDIV           equ     6       ;Division
foCMP           equ     8       ;compare
foCPX           equ     10      ;compare
foREM           equ     12      ;remainder
foZ2X           equ     14      ;->étendu
foX2Z           equ     16      ;étendu->
foSQRT          equ     18      ;racine carrée
foRTI           equ     20      ;round  to integral
foTTI           equ     22      ;truncate to integral
foSCALB         equ     24      ;binary scale
foLOGB          equ     26      ;logarithme base 2
foCLASS         equ     28      ;classify
; UNDEFINED     equ     30
foSETENV        equ     1       ;set environment
foGETENV        equ     3       ;get environment
foSETHV         equ     5       ;set halt vector
foGETHV         equ     7       ;get halt vector
foD2B           equ     9       ;décimal->binaire
foB2D           equ     11      ;binaire->décimal
foNEG           equ     13      ;negate
foABS           equ     15      ;valeur absolue
foCPYSGN        equ     17      ; copy sign
foNEXT          equ     19      ; next-after
foSETXCP        equ     21      ; set exception
foPROCENTRY     equ     23      ; procedure entry
foPROCEXIT      equ     25      ; procedure exit
foTESTXCP       equ     27      ; test exception
; UNDEFINED     equ     29
; UNDEFINED     equ     31
* NUMEROS DES FORMATS D'OPERANDES ---------------------
ffEXT           equ     $0000
ffDBL           equ     $0800
ffSGL           equ     $1000
ffINT           equ     $2000
ffLNG           equ     $2800
ffcOMP          equ     $3000
*------------------------------------------------------
* Precision code masks: forces a floating point
* output value  to be coerced to the range and
* precision specified.
*------------------------------------------------------
fcEXT           equ     $0000   ;étendu
fcDBL           equ     $4000   ;double
fcSGL           equ     $8000   ;single
*------------------------------------------------------
;------------------------------------------------------
; Class and sign inquiries.
;------------------------------------------------------
fcSNAN          equ     1       ; signaling NAN
fcQNAN          equ     2       ; quiet NAN
fcINf           equ     3       ; infinity
fcZERO          equ     4       ; zero
fcNORM          equ     5       ; normal number
fcDENORM        equ     6       ; denormal number
;------------------------------------------------------
; Bit indexes for bytes of floating point environment
; word.
;------------------------------------------------------
fBINVALID       equ     0 ; invalid operation
fBUfLOW         equ     1 ; underflow
fBOfLOW         equ     2 ; overflow
fBDiv2ER        equ     3 ; Division by zero
fBINEXACT       equ     4 ; inexact
fBRNDLO         equ     5 ; low bit of rounding
fBRNDHI         equ     6 ; high bit of rounding
fBLSTRND        equ     7 ; last round result bit
fBDBL           equ     5 ; double precision control
fBSGL           equ     6 ; single precision control
* Numéros des fonctions--------------------------------
foLNX           equ     $0000   ;logarithme néperien
foLOG2X         equ     $0002   ;logarithme de base 2
foLN1X          equ     $0004   ;Lg(1+x)
```

*Suite page 54…*

# Essai Orthogiciel 2



Orthogiciel Plus de Larousse est une nouvelle version du correcteur orthographique pour Macintosh. Elle intègre de nouvelles possibilités telle la correction de documents 'entiers' sauvegardés par MacWrite 4.5 et 2.0.

## Orthogiciel

Il est livré sur deux disquettes contenues dans un petit classeur mode d'emploi bien présenté. Pas de difficultés de mise en route, peu de contraintes d'installation, aucun problème de mise en place sur disque dur.

Le principe retenu pour la correction est le suivant : un document a été sauvegardé par MacWrite 4.5 ou 2.0 (ou par tout autre traitement de textes en format 'texte seul') ;

- lancer Orthogiciel,
- ouvrir le document,
- demander la vérification.

Orthogiciel s'arrête sur chaque mot posant un problème et propose de le maintenir en l'état, de le corriger, de consulter le dictionnaire avant correction.

Le dictionnaire sur disquette ne comprend pas tout le petit Larousse mais 150 000 formes ce qui est beaucoup pour une disquette mais pas trop pour le français.

## La grammaire

Orthogiciel est un correcteur orthographique et à ce titre, il laisse de côté les erreurs de syntaxe et fautes d'accord. On pourra écrire «nous avez» sans émouvoir le programme (déceler ce type de faute est d'une difficulté sans commune mesure avec la comparaison à un dictionnaire, même de 150 000 mots).

Le mode d'emploi, pour compenser cette restriction propose un guide d'orthographe qui aide à ne pas accorder «elle s'est complu» et à accorder «elle s'est abstenue» par exemple.

Le programme propose aussi une aide à la conjugaison des verbes qui connaît les formes défectives éventuelles, les doubles formes et les participes invariables (du type succédé).

## Le dictionnaire

Surtout incomplet, le petit Larousse ne permet pas de satisfaire les besoins de chacun. Orthogiciel permet de définir un ou plusieurs dictionnaires utilisateur pour y inclure le jargon de telle ou telle profession. Chaque dictionnaire additionnel peut contenir 500 mots et on ne peut utiliser qu'un seul de ces dictionnaires à la fois.

## Et les accents ?

Orthogiciel, bien que conçu pour le Macintosh se plante systématiquement si la zélée secrétaire prend le soin d'écrire «États-Unis, œuvre, Œuf, Âtre» au lieu de «Etats-Unis, oeuvre, Oeuf, Atre».
Idem pour les ligatures ﬁ et ﬂ : le programme préfère nettement «fichier» à «ﬁchier». C'est bien regrettable car, bien utilisé, le Mac produit une typographie de qualité douée de Ç et autres Ô. «Ça marche !» ne marche justement pas.

## Mots défectifs

Tout le français ne tient pas sur un disque et le choix n'a pas dû être aisé. Toutefois, des mots d'usage fréquent manquent à l'appel :

logo, convivial, décodeur, positionnement, picots, réutilisé.

Le programme s'arrête également sur surnatalité, numérisée, crénage, enfiche, occlusives, salvateur, compilez...

Générer est bien dans le dictionnaire Orthogiciel mais les formes conjuguées généré, génère, générant ne sont pas reconnues.

Plus curieuse est la scission aléatoire de certains mots : procédures, position, caractères sont parfois reconnus comme pro, cédures, po, sition, ca, ractères... d'où arrêts fréquents et inutiles de la vérification.

Fichiers Vérification Options Dictionnaires Boîte à outils

orthogiciel

A l'usage
Doté de la fonction 'renvoyer', sorte de copier/coller,
Orthogiciel est souple d'emploi. Un menu propose l'aide à la

Correction

Il est assez rapide et même très rapide en considérant l'importance de la recherche.

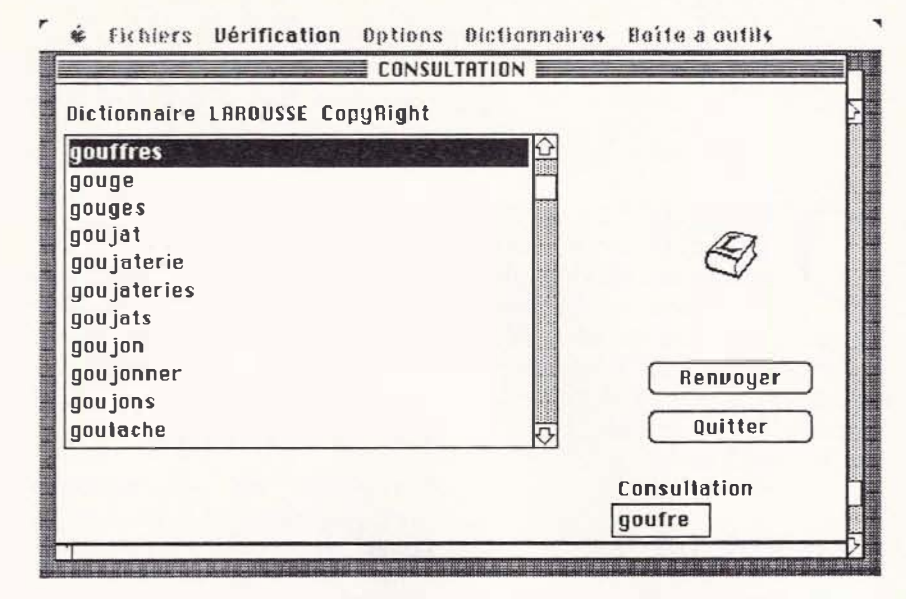
Consultation du Larousse | l'impor-
Aide à la correction | Remplacer | Maintenir
Pluriel des noms | Ajouter au dictionnaire
Conjugaison des verbes | Arrêter la Correction

En cas de césure avec trait d'union comme ci-dessus, le mot union n'est pas reconnu, mais gêne est minime.

## À l'usage

Doté de la fonction 'renvoyer', sorte de copier/coller, Orthogiciel est souple d'emploi. Un menu propose l'aide à la conjugaison, un générateur de pluriel et l'aide à la correction par une recherche simplifiée dans le dictionnaire.
Il est assez rapide et même très rapide en considérant l'importance de la recherche.

Le défaut le plus pénible à supporter apparaît lors de la détection d'une faute :
une fenêtre s'affiche, avec un extrait de la phrase, le mot en cause et neuf boutons de contrôle. Le drame est que cette fenêtre disparaît et réapparraît à chaque détection d'où un fatigue visuelle dès la 5ème faute et la migraine à la 10ème. Pourquoi ne pas avoir opté pour deux fenêtres, en haut et en bas de l'écran, affichées simultanément ?




Le confort y aurait gagné. À noter que les fenêtres sont dotées de barres de déplacement alors qu'on ne peut les déplacer.

## En conclusion

Des bons points : le prix, la rapidité, l'efficacité, la possibilité de l'installer sur disque dur : la disquette originale n'est demandée que de temps en temps.

Des points noirs : cette fenêtre qui 'flashe', les œ É et autres Ô ignorés, de petits défauts de fonctionnement, l'impossibilité de faire accepter 'CP/M' même à l'aide du dictionnaire utilisateur du fait du '/'.

Orthogiciel reste toutefois irremplaçable pour des fautes du type «programmme» qui peuvent échapper à la relecture la plus attentive.

```
foLOG21X        equ     $0006   ;log2(1+x)
foEXPX          equ     $0008 ;exponentielle de base e
foEXP2X         equ     $000A ;exponentielle de base 2
foEXP1X         equ     $000C   ;exp(x)-1
foEXP21X        equ     $000E   ;exp2(x)-1
foXPWRI         equ     $8010 ;exponentiation entière
foXPWRY         equ     $8012 ;exponentiation générale
foCOMPOUND      equ     $C014   ;compound
foANNUITY       equ     $C016   ;annuity
foSINX          equ     $0018   ;sinus
foCOSX          equ     $001A   ;cosinus
foTANX          equ     $001C   ;tangente
foATANX         equ     $001E   ;Arctangente
foRANDX         equ     $0020   ;nombre aléatoire
;------------------------------------------------------
; NaN codes.
;------------------------------------------------------
NANSQRT         equ     1
NANAdd          equ     2
NANDiv          equ     4
NANMul          equ     8
NANREM          equ     9
NANASCBIN       equ     17
NANCOMP         equ     20
NANZERO         equ     21
NANTRIG         equ     33
NANINVTRIG      equ     34
NANLOG          equ     36
NANPOWER        equ     37
NANfINAN        equ     38
NANINIT         equ     255
```

*...suite de la page 52*

# COPY : une commande externe

F. Rozay

Ce programme ajoute une commande COPY à ProDOS. En effet cette commande non implantée d'origine lui fait cruellement défaut car il faut avouer qu'il est plus pratique de donner un ordre directement au clavier plutôt que sortir son copieur favori.

Cette commande servira pour recopier un fichier texte volumineux d'un disque à un autre par exemple.

## Mode d'emploi

Cette nouvelle commande est disponible au clavier ou par programme après avoir tapé :

```
BRUN COPY
```
ou
```
- COPY
```

sous ProDOS exclusivement.

La syntaxe de la commande est la même que celle des autres commandes ProDOS, à savoir :

```
COPY chemin1, chemin2
```

Chemin1 est le chemin d'accès (pathname) du fichier source, chemin2 celui du fichier de destination, par exemple :

```
COPY /bureau/dossier/
client,/archive/client
```

Si le nom du volume est omis, le programme prendra le lecteur par défaut, c'est à dire celui que ProDOS utilise quand on tape CATALOG.

Cette commande externe permet la copie de tous types de fichiers (system, text, AppleWorks, binaires, commande...) mais ne peut copier des volumes. Ainsi, l'ordre :

```
COPY /DISQUE/,/RAM5/
```

est invalide.

Si l'utilisateur ne possède qu'un lecteur de disquettes, il devra utiliser l'éventuel disque virtuel comme tampon.



## Programme COPY

Après avoir saisi ce code sous moniteur, vous le sauvegarderez par :

```
BSAVE COPY, A$6000,L$A08
```

```
6000:AD 08 BE 8D 62 61 AD 07
6008:BE 8D 61 61 A9 0A 8D BA
6010:60 20 F5 BE C9 0C D0 03
6018:4C 09 BE 8D 08 BE 85 FB
6020:A8 AE BA 60 8A 48 98 48
6028:29 07 A8 B9 BB 60 AA 68
6030:48 4A 4A 4A A8 8A 19 58
6038:BF 99 58 BF 68 A8 C8 68
6040:AA CA D0 E0 AD BA 60 38
6048:ED FB BE 8D FB BE A9 00
6050:8D 07 BE 85 FA A9 61 85
6058:3B A9 00 85 3A 20 62 60
6060:18 60 A2 00 A0 FF A1 3A
6068:F0 31 20 8C F8 A4 2F C0
6070:02 D0 28 B1 3A C9 BF D0
6078:10 88 B1 3A C9 00 D0 09
6080:88 B1 3A A0 05 C9 20 F0
6088:02 A0 02 B1 3A 38 E9 61
6090:90 09 C9 0A B0 05 6D 08
6098:BE 91 3A A2 00 C8 A1 3A
60A0:81 FA E6 3A E6 FA D0 0A
60A8:E6 3B E6 FB A9 6B C5 3B
60B0:F0 07 88 30 E6 D0 E7 F0
60B8:A9 60 00 80 40 20 10 08
60C0:04 02 01 00 00 00 00 00
60C8:00 00 00 00 00 00 00 00
60D0:00 00 00 00 00 00 00 00
60D8:00 00 00 00 00 00 00 00
60E0:00 00 00 00 00 00 00 00
60E8:00 00 00 00 00 00 00 00
60F0:00 00 00 00 00 00 00 00
60F8:00 00 00 00 00 00 00 00
6100:D8 A6 06 AD 6C BE 85 06
6108:A4 07 AD 6D BE 85 07 8E
6110:68 65 8C 69 65 A2 00 A0
6118:01 B1 06 C9 A0 D0 04 C8
6120:4C 19 61 EC 6C 65 F0 0A
6128:DD 6D 65 D0 28 E8 C8 4C
6130:19 61 88 88 8C 52 BE AD
6138:63 61 AD 38 61 AC 39 61
6140:8D 50 BE 8C 51 BE A9 00
6148:8D 53 BE 8D 55 BE A9 03
6150:8D 54 BE 18 60 38 AD 68
6158:65 AC 69 65 85 06 84 07
6160:4C FF FF AD 56 BE 29 03
6168:C9 03 F0 05 A9 10 4C 09
6170:BE AD 6C BE AC 6D BE 8D
6178:72 65 8C 73 65 AD 6E BE
6180:AC 6F BE 8D 92 65 8C 93
6188:65 A5 73 A4 74 8D 64 65
6190:8C 65 65 AD 6C BE AC 6D
6198:BE 85 06 84 07 A0 01 B1
61A0:06 C9 2F F0 03 20 62 63
61A8:AD 6E BE AC 6F BE 85 06
61B0:84 07 A0 01 B1 06 C9 2F
61B8:F0 03 20 62 63 20 2E 64
61C0:AD 75 65 C9 01 F0 04 C9
61C8:0F D0 05 A9 0D 4C 09 BE
61D0:A2 02 E8 BD 71 65 9D 91
61D8:65 E0 07 D0 F5 A9 E3 8D
61E0:94 65 20 37 64 AD 90 BF
61E8:AE 91 BF 8D 99 65 8E 9A
61F0:65 AD 92 BF AE 93 BF 8D
61F8:9B 65 8E 9C 65 20 40 64
6200:A9 04 20 9A 64 A0 00 8C
6208:B0 65 8D B1 65 AD 6C BE
6210:AC 6D BE 8D AE 65 8C AF
6218:65 20 49 64 AD B2 65 8D
6220:A6 65 8D B4 65 A9 04 20
6228:9A 64 A0 00 8C B0 65 8D
6230:B1 65 AD 6E BE AC 6F BE
6238:8D AE 65 8C AF 65 20 49
6240:64 AD B2 65 8D 9E 65 20
6248:64 64 A9 00 8D 5E 65 8D
6250:5F 65 8D 60 65 AD 79 65
6258:AC 7A 65 8D 66 65 8C 67
6260:65 0E 66 65 2E 67 65 AD
6268:67 65 F0 07 A9 8F 8D 6B
6270:65 D0 07 AE 66 65 E8 8E
6278:6B 65 CE 6B 65 AD 6B 65
6280:20 F5 BE B0 F5 8D A8 65
6288:8D A0 65 A0 00 8C A7 65
6290:8C 9F 65 AD 6B 65 8D 6A
6298:65 AD B5 65 38 ED 5E 65
62A0:8D 61 65 AD B6 65 ED 5F
62A8:65 8D 62 65 AD B7 65 ED
62B0:60 65 8D 63 65 AD 63 65
62B8:D0 14 AD 62 65 CD 6A 65
62C0:B0 0C 8D AA 65 AD 61 65
62C8:8D A9 65 4C D9 62 A9 00
62D0:AC 6A 65 8D A9 65 8C AA
62D8:65 20 6D 64 AE AB 65 AC
62E0:AC 65 8A 18 6D 5E 65 8D
62E8:5E 65 98 6D 5F 65 8D 5F
62F0:65 A9 00 6D 60 65 8D 60
```

```
62F8:65 8E A1 65 8C A2 65 20
6300:52 64 AD 60 65 CD B7 65
6308:90 10 AD 5F 65 CD B6 65
6310:90 08 AD 5E 65 CD B5 65
6318:F0 03 4C 99 62 20 3B 63
6320:AD 6E BE AC 6F BE 8D 84
6328:65 8C 85 65 A2 04 BD 73
6330:65 9D 85 65 CA D0 F7 20
6338:91 64 60 AD A6 65 8D C3
6340:65 20 5B 64 AD 9E 65 8D
6348:C3 65 20 5B 64 AD 64 65
6350:AC 65 65 85 73 84 74 AD
6358:68 65 AC 69 65 85 06 84
6360:07 60 A9 01 20 9A 64 A0
6368:00 8D BB 65 8C BA 65 8D
6370:BE 65 8C BD 65 8D 2A 64
6378:8C 29 64 20 76 64 20 28
6380:64 F0 01 60 AD 3C BE 0A
6388:0A 0A 0A 0A AA AC 3D BE
6390:88 98 4A 8A 6A 8D B9 65
6398:20 7F 64 A5 06 18 69 01
63A0:85 3C A5 07 69 00 85 3D
63A8:A0 00 B1 06 38 65 06 85
63B0:3E A5 07 69 00 85 3F 20
63B8:28 64 18 69 03 18 65 06
63C0:85 42 A5 07 69 00 85 43
63C8:A5 3E 38 E5 3C A8 C8 88
63D0:B1 3C 91 42 C0 00 D0 F7
63D8:AD BA 65 18 69 01 85 3C
63E0:AD BB 65 69 00 85 3D 20
63E8:28 64 38 6D BA 65 85 3E
63F0:AD BB 65 69 00 85 3F A5
63F8:06 18 69 02 85 42 A5 07
6400:69 00 85 43 20 2C FE A0
6408:00 B1 06 8D 13 64 20 28
6410:64 18 69 00 69 02 91 06
6418:C8 A9 AF 91 06 20 28 64
6420:18 69 02 A8 A9 AF 91 06
6428:AD FF FF 29 0F 60 20 00
6430:BF C4 71 65 B0 6A 60 20
6438:00 BF 82 00 00 B0 61 60
6440:20 00 BF C0 91 65 B0 58
6448:60 20 00 BF C8 AD 65 B0
6450:4F 60 20 00 BF CB 9D 65
6458:B0 46 60 20 00 BF CC C2
6460:65 B0 3D 60 20 00 BF D1
6468:B3 65 B0 34 60 20 00 BF
6470:CA A5 65 B0 2B 60 20 00
6478:BF C7 BC 65 B0 22 60 20
6480:00 BF C5 B8 65 B0 19 60
6488:20 00 BF C1 BF 65 B0 10
6490:60 20 00 BF C3 83 65 B0
6498:07 60 20 F5 BE B0 01 60
64A0:48 C9 48 D0 12 AD 92 65
64A8:AC 93 65 8D C0 65 8C C1
64B0:65 20 3B 63 20 88 64 AD
64B8:C4 65 AD B8 64 AC B9 64
64C0:20 3A DB 68 48 20 DA FD
64C8:A9 A0 20 ED FD A9 BA 20
64D0:ED FD A9 A0 20 ED FD 68
64D8:A2 1C DD CD 65 F0 03 CA
64E0:D0 F8 8A 8D E9 64 0A 18
64E8:69 00 AA E8 BD 06 65 E8
64F0:BC 06 65 20 3A DB A9 07
64F8:20 ED FD A9 8D 20 ED FD
6500:20 4D 63 4C D0 03 AD EA
6508:65 AD 0B 66 AD 2B 66 AD
6510:54 66 AD 6C 66 AD 8A 66
6518:AD A7 66 AD C3 66 AD E5
```

```
6520:66 AD 08 67 AD 44 67 AD
6528:5F 67 AD 7F 67 AD AF 67
6530:AD C7 67 AD E9 67 AD 0B
6538:68 AD 38 68 AD 68 68 AD
6540:8A 68 AD B9 68 AD D4 68
6548:AD F1 68 AD 0D 69 AD 2C
6550:69 AD 59 69 AD 8B 69 AD
6558:BE 69 AD E5 69 00 00 00
6560:00 00 00 00 00 00 00 00
6568:00 00 00 00 04 43 4F 50
6570:59 0A 00 00 00 00 00 00
6578:00 00 00 00 00 00 00 00
6580:00 00 00 07 00 00 00 00
6588:00 00 00 00 00 00 00 00
6590:00 07 00 00 00 00 00 00
6598:00 00 00 00 00 04 00 00
65A0:00 00 00 00 00 04 00 00
65A8:00 00 00 00 00 03 00 00
65B0:00 00 00 02 00 00 00 00
65B8:02 00 00 00 01 00 00 01
65C0:00 00 01 00 8D A0 C3 EF
65C8:E4 E5 A0 A4 00 01 04 25
65D0:27 28 2B 2E 40 42 43 44
65D8:45 46 47 48 49 4A 4B 4C
65E0:4D 4E 50 51 52 53 55 56
65E8:57 5A CD E1 F5 F6 E1 E9
65F0:F3 A0 E3 EF E4 E5 A0 E4
65F8:E5 A0 E6 EF EE E3 F4 E9
6600:EF EE A0 E4 F5 A0 CD CC
6608:C9 AE 00 CE EF ED E2 F2
6610:E5 A0 E4 E5 A0 F0 E1 F2
6618:E1 ED FD F4 F2 E5 F3 A0
6620:E9 EE E3 EF F2 F2 E5 E3
6628:F4 AE 00 D4 E1 E2 EC E5
6630:A0 E4 E5 A0 F6 E5 E3 F4
6638:E5 F5 F2 F3 A0 E4 A7 E9
6640:EE F4 E5 F2 F2 F5 F0 F4
6648:E9 EF EE A0 F0 EC E5 E9
6650:EE E5 AE 00 C5 F2 F2 E5
6658:F5 F2 A0 E4 A7 E5 EE F4
6660:F2 FB E5 AF F3 EF F2 F4
6668:E9 E5 AE 00 D0 E1 F3 A0
6670:E4 E5 A0 F0 FB F2 E9 F0
6678:E8 FB F2 E9 F1 F5 E5 A0
6680:E3 EF EE EE E5 E3 F4 FB
6688:AE 00 C4 E9 F3 F1 F5 E5
6690:A0 F0 F2 EF F4 FB E7 FB
6698:A0 C0 A0 EC A7 FB E3 F2
66A0:E9 F4 F5 F2 E5 AE 00 CC
66A8:E1 A0 E4 E9 F3 F1 F5 E5
66B0:F4 F4 E5 A0 E1 A0 FB F4
66B8:FB A0 E3 E8 E1 EE E7 FB
66C0:E5 AE 00 CC E1 A0 F3 F9
66C8:EE F4 E1 F8 E5 A0 E4 F5
66D0:A0 EE EF ED A0 E5 F3 F4
66D8:A0 E9 EE E3 EF F2 F2 E5
66E0:E3 F4 E5 AE 00 C9 EC A0
66E8:F9 A0 E1 A0 E4 FB EA E1
66F0:A0 E8 F5 E9 F4 A0 E6 E9
66F8:E3 E8 E9 E5 F2 F3 A0 EF
6700:F5 F6 E5 F2 F4 F3 AE 00
6708:C3 E5 A0 EE F5 ED FB F2
6710:EF A0 E4 E5 A0 F2 FB E6
6718:FB F2 E5 EE E3 E5 A0 EE
6720:A7 E5 F3 F4 A0 F0 E1 F3
6728:A0 E3 E5 EC F5 E9 A0 E4
6730:A7 F5 EE A0 E6 E9 E3 E8
6738:E9 E5 F2 A0 EF F5 F6 E5
6740:F2 F4 AE 00 D3 EF F5 F3
```

```
6748:AD E3 E1 F4 E1 EC EF E7
6750:F5 E5 A0 E9 EE E5 F8 E9
6758:F3 F4 E1 EE F4 AE 00 CC
6760:E5 A0 F6 EF EC F5 ED E5
6768:A0 EE A7 E5 F3 F4 A0 F0
6770:E1 F3 A0 E4 E9 F3 F0 EF
6778:EE E9 E2 EC E5 AE 00 CC
6780:E5 A0 E6 E9 E3 E8 E9 E5
6788:F2 A0 EE A7 E5 F3 F4 A0
6790:F0 E1 F3 A0 E4 E1 EE F3
6798:A0 EC E5 A0 E3 E1 F4 E1
67A0:EC EF E7 F5 E5 A0 E4 E5
67A8:ED E1 EE E4 FB AE 00 CC
67B0:E5 A0 E6 E9 E3 E8 E9 E5
67B8:F2 A0 E5 F8 E9 F3 F4 E5
67C0:A0 E4 FB EA C0 AE 00 D0
67C8:E1 F3 A0 E1 F3 F3 E5 FA
67D0:A0 E4 E5 A0 F0 EC E1 E3
67D8:E5 A0 F3 F5 F2 A0 E3 E5
67E0:A0 F6 EF EC F5 ED E5 AE
67E8:00 CC E5 A0 E3 E1 F4 E1
67F0:EC EF E7 F5 E5 A0 F0 F2
67F8:E9 EE E3 E9 F0 E1 EC A0
6800:E5 F3 F4 A0 F0 EC E5 E9
6808:EE AE 00 CC E5 A0 CD CC
6810:C9 A0 E5 F3 F4 A0 F4 F2
6818:EF F0 A0 E1 EE E3 E9 E5
6820:EE A0 F0 EF F5 F2 A0 EC
6828:E9 F2 E5 A0 E3 E5 A0 E6
6830:E9 E3 E8 E9 E5 F2 AE 00
6838:C3 E5 A0 D0 F2 EF C4 CF
6840:D3 A0 EE E5 A0 F0 E5 F5
6848:F4 A0 F5 F4 E9 EC E9 F3
6850:E5 F2 A0 E3 E5 A0 F4 F9
6858:F0 E5 A0 E4 E5 A0 F3 F4
6860:EF E3 EB E1 E7 E5 AE 00
6868:CC E1 A0 E6 E9 EE A0 E4
6870:F5 A0 E6 E9 E3 E8 E9 E5
6878:F2 A0 E1 A0 FB F4 FB A0
6880:E1 F4 F4 E5 E9 EE F4 E5
6888:AE 00 CF EE A0 EE E5 A0
6890:F0 E5 F5 F4 A0 E1 EC EC
6898:E5 F2 A0 E1 F5 A0 E4 E5
68A0:EC C0 A0 E4 E5 A0 EC E1
68A8:A0 E6 E9 EE A0 E4 F5 A0
68B0:E6 E9 E3 E8 E9 E5 F2 AE
68B8:00 CC E5 A0 E6 E9 E3 E8
68C0:E9 E5 F2 A0 E5 F3 F4 A0
68C8:F6 E5 F2 F2 EF F5 E9 EC
68D0:EC FB AE 00 CC E5 A0 E6
68D8:E9 E3 E8 E9 E5 F2 A0 E5
68E0:F3 F4 A0 F2 E5 F3 F4 FB
68E8:A0 EF F5 F6 E5 F2 F4 AE
68F0:00 CC E5 A0 E3 E1 F4 E1
68F8:EC EF E7 F5 E5 A0 E5 F3
6900:F4 A0 E5 EE E4 EF ED ED
6908:E1 E7 FB AE 00 CC E5 A0
6910:E4 E9 F3 F1 F5 E5 A0 EE
6918:A7 E5 F3 F4 A0 F0 E1 F3
6920:A0 E5 EE A0 D0 F2 EF C4
6928:CF D3 AE 00 D5 EE A0 E4
6930:E5 F3 A0 F0 E1 F2 E1 ED
6938:FD F4 F2 E5 F3 A0 E4 E5
6940:A0 EC E1 A0 F4 E1 E2 EC
6948:E5 A0 E5 F3 F4 A0 E9 EE
6950:E3 EF F2 F2 E5 E3 F4 AE
6958:00 D0 F2 EF C4 CF D3 A0
6960:EE E5 A0 F0 E5 F5 F4 A0
6968:E7 E5 F2 E5 F2 A0 F1 F5
```

```
6970:E5 A0 E8 F5 E9 F4 A0 EC
6978:E5 E3 F4 E5 F5 F2 F3 A0
6980:C0 A0 EC E1 A0 E6 EF E9
6988:F3 AE 00 CC E5 A0 E2 F5
6990:E6 E6 E5 F2 A0 EE E5 A0
6998:F0 E5 F5 F4 A0 F3 A7 E9
69A0:ED F0 EC E1 EE F4 E5 F2
69A8:A0 C0 A0 EC A7 E5 EE E4
69B0:F2 EF E9 F4 A0 E4 FB F3
69B8:E9 E7 EE FB AE 00 C9 EC
69C0:A0 F9 A0 E1 A0 E4 E5 F5
69C8:F8 A0 EE EF ED F3 A0 E4
69D0:E5 A0 F6 EF EC F5 ED E5
69D8:A0 E9 E4 E5 EE F4 E9 F1
69E0:F5 E5 F3 AE 00 CC E1 A0
69E8:E3 E1 F2 F4 E5 A0 E4 E5
69F0:F3 A0 E2 EC EF E3 F3 A0
69F8:E5 F3 F4 A0 E5 EE E4 EF
6A00:ED ED E1 E7 FB E5 AE 00
```

## Source COPY.S
### Assembleur Merlin Pro

//e
//e+
//c
][gs

```
**********
*
* Copie de fichiers ProDOS
*
* Frédéric Rozay / juillet-aout 87
*
**********

         LST   OFF

BUFFER   =     $6
A1       =     $3C
A2       =     $3E
A4       =     $42

START    =     $6000
DEBUT    =     START+$100

LENGTH   =     $2F
PCL      =     $3A
HIMEM    =     $73

ADR      =     $FA

WARM     =     $3D0

EXTRNCMD =     $BE07
ERROUT   =     $BE09
DEFSLT   =     $BE3C
DEFDRV   =     $BE3D
XTERNADDR =    $BE50
XLEN     =     $BE52
XCNUM    =     $BE53
PBITS    =     $BE54
FBITS    =     $BE56
VPATH1   =     $BE6C
VPATH2   =     $BE6E
HIMMSB   =     $BEFB
GETBUFR  =     $BEF5
MLI      =     $BF00
BITMAP   =     $BF58
DATE     =     $BF90
TIME     =     $BF92

STROUT   =     $DB3A

INSDSP2  =     $F88C
PRBYTE   =     $FDDA
COUT     =     $FDED
MOVE     =     $FE2C

GET_TIME =     $82
DESTROY  =     $C1
CREATE   =     $C0
SET_FILE_INFO = $C3
GET_FILE_INFO = $C4
ONLINE   =     $C5
GET_PREFIX =   $C7
OPEN     =     $C8
READ     =     $CA
WRITE    =     $CB
CLOSE    =     $CC
GET_EOF  =     $D1

         ORG   START
 Relogeur de le commande
 externe

         LDA   EXTRNCMD+1
         STA   OLDCMD+2
         LDA   EXTRNCMD
         STA   OLDCMD+1

         LDA   £>PFIN-DEBUT+$FF
         STA   NBPAGES
         JSR   GETBUFR
         CMP   £$0C
         BNE   GOTHEM
         JMP   ERROUT

GOTHEM   STA   EXTRNCMD+1
         STA   ADR+1
         TAY
         LDX   NBPAGES

 note les pages utilisées
 pour abaisser Himem
MARKUSED TXA
         PHA
         TYA
         PHA
         AND   £7
         TAY
         LDA   BIT.MASK,Y
         TAX
         PLA
         PHA
         LSR
         LSR
         LSR
         TAY
         TXA
         ORA   BITMAP,Y
         STA   BITMAP,Y
         PLA
         TAY
         INY
         PLA
         TAX
         DEX
         BNE   MARKUSED

         LDA   NBPAGES
         SEC
         SBC   HIMMSB
         STA   HIMMSB

         LDA   £0
         STA   EXTRNCMD
         STA   ADR
         LDA   £>DEBUT
         STA   PCL+1
         LDA   £<DEBUT
         STA   PCL
         JSR   RELOGE
         CLC
         RTS

* Relogeur
* réassemble le programme
  situé à l'adresse (ADRESSE)
* DEBUT doit etre multiple de 256
* la zone des données est
  séparée du programme par un 00
* le programme reloge la
  structure du MLI
* ainsi que les parametres
  éventuels

RELOGE   LDX   £0
         LDY   £$FF
         LDA   (PCL,X)
         BEQ   RTRANS
         JSR   INSDSP2
         LDY   LENGTH
         CPY   £2
         BNE   RTRANS
         LDA   (PCL),Y
         CMP   £$BF
         BNE   NOMLI
         DEY
         LDA   (PCL),Y
         CMP   £0
         BNE   NOMLI
         DEY
         LDA   (PCL),Y
         LDY   £5
         CMP   £$20
         BEQ   RMLI
NOMLI    LDY   £2
RMLI     LDA   (PCL),Y
         SEC
         SBC   £>DEBUT
         BCC   RTRANS
         CMP   £>PFIN-DEBUT+$FF
         BCS   RTRANS
         ADC   EXTRNCMD+1
         STA   (PCL),Y

RTRANS   LDX   £0
         INY
RBOUCLE  LDA   (PCL,X)
         STA   (ADR,X)
         INC   PCL
         INC   ADR
         BNE   NOCARRY
         INC   PCL+1
         INC   ADR+1
         LDA   £>PFIN+$FF
         CMP   PCL+1
         BEQ   RFIN
NOCARRY  DEY
         BMI   RTRANS
         BNE   RBOUCIE
         BEQ   RELOGE

RFIN     RTS

NBPAGES  DS    1
BITMASK  DFB   $80,$40,$20,$10,8,4,2,1

* Vérifie que la commande
* est bien COPY

         DS    DEBUT-*

         CLD
         LDX   BUFFER
         LDA   VPATH1
         STA   BUFFER
         LDY   BUFFER+1
         LDA   VPATH1+1
         STA   BUFFER+1
         STX   OLDBUFF
         STY   OLDBUFF+1
         LDX   £0
         LDY   £1
TEST     LDA   (BUFFER),Y
         CMP   £" "
         BNE   TEST1
         INY
         JMP   TEST
TEST1    CPX   COMMAND
         BEQ   MIENNE
         CMP   COMMAND+1,X
         BNE   PASMIEN
         INX
         INY
         JMP   TEST

MIENNE   DEY
         DEY
         STY   XLEN
MIENNE1  LDA   COPY
         LDA   MIENNE1+1
         LDY   MIENNE1+2
         STA   XTERNADDR
         STY   XTERNADDR+1
         LDA   £0
         STA   XCNUM
         STA   PBITS+1
         LDA   £3
         STA   PBITS
         CLC
         RTS

PASMIEN  SEC
         LDA   OIDBUFF
         LDY   OLDBUFF+1
         STA   BUFFER
         STY   BUFFER+1
OLDCMD   JMP   $FFFF

**********
*
* Initialisation pour la copie
*
**********

 deux pathnames nécessaires
COPY     LDA   FBITS
         AND   £3
         CMP   £3
         BEQ   GOODBOY

         LDA   £$I0
         JMP   ERROUT

GOODBOY  LDA   VPATH1
         LDY   VPATH1+1
         STA   PARMGFI+1
         STY   PARMGFI+2

         LDA   VPATH2
         LDY   VPATH2+1
         STA   PARMCRE+1
         STY   PARMCRE+2

         LDA   HIMEM
         LDY   HIMEM+1
         STA   OLDHIM
         STY   OLDHIM+1

         LDA   VPATH1
         LDY   VPATH1+1
         STA   BUFFER
         STY   BUFFER+1
         LDY   £1
         LDA   (BUFFER),Y
         CMP   £'/'
         BEQ   PRFXST1

         JSR   INCPATH

PRFXST1  LDA   VPATH2
         LDY   VPATH2+1
         STA   BUFFER
         STY   BUFFER+1
         LDY   £1
         LDA   (BUFFER),Y
         CMP   £'/'
```

```
         BEQ   PRFXST2

         JSR   INCPATH

PRFXST2  JSR   GETFINFO

* Vérifie que le fichier
 n'est ni un Directory ni
 un fichier Bad

         LDA   PARMGFI+4
         CMP   £1
         BEQ   BADTYPE
         CMP   £$F
         BNE   GOODTYPE

BADTYPE  LDA   £$D
         JMP   ERROUT

* Recopie les paramètres
 du fichier source dans
 la table des
* paramètres du fichier
 destination

GOODTYPE LDX   £2
GFI_CRE  INX
         LDA   PARMGFI,X
         STA   PARMCRE,X
         CPX   £7
         BNE   GFI_CRE
         LDA   £%11100011
         STA   PARMCRE+3

* Va chercher l'heure et
 la date de création
* du fichier destination

         JSR   HEUDAT

         LDA   DATE
         LDX   DATE+1
         STA   PARMCRE+8
         STX   PARMCRE+9
         LDA   TIME
         LDX   TIME+1
         STA   PARMCRE+10
         STX   PARMCRE+11

         JSR   CREE

* Ouvre le fichier source
 puis le fichier destination

         LDA   £4
         JSR   GBUFF
         LDY   £0
         STY   PARMOUV+3
         STA   PARMOUV+4

         LDA   VPATH1
         LDY   VPATH1+1
         STA   PARMOUV+1
         STY   PARMOUV+2

         JSR   OUVRE

         LDA   REFNUM
         STA   PARMLIT+1
         STA   PARMEOF+1

         LDA   £4
         JSR   GBUFF
         LDY   £0
         STY   PARMOUV+3
         STA   PARMOUV+4

         LDA   VPATH2
         LDY   VPATH2+1
         STA   PARMOUV+1
         STY   PARMOUV+2

         JSR   OUVRE

         LDA   REFNUM
         STA   PARMECR+1

         JSR   GEOF

* Choisit un buffer pour
 le transfert

         LDA   £0
         STA   POSEOF
         STA   POSEOF+1
         STA   POSEOF+2

         LDA   PARMGFI+8
         LDY   PARMGFI+9
         STA   CALCUL
         STY   CALCUL+1
         ASL   CALCUL
         ROL   CALCUL+1
         LDA   CALCUL+1
         BEQ   TAILLE
         LDA   £143
         STA   ECCO
         BNE   CHERBUF
TAILLE   LDX   CALCUL
         INX
         STX   ECCO
CHERBUF  DEC   ECCO
         LDA   ECCO
         JSR   GETBUFR
         BCS   CHERBUF
         STA   PARMLIT+3
         STA   PARMECR+3
         LDY   £0
         STY   PARMLIT+2
         STY   PARMECR+2
         LDA   ECCO
         STA   LENBUFF

* Procède au transfert
 des informations

SUITE    LDA   PARMEOF+2
         SEC
         SBC   POSEOF
         STA   RESTE
         LDA   PARMEOF+3
         SBC   POSEOF+1
         STA   RESTE+1
         LDA   PARMEOF+4
         SBC   POSEOF+2
         STA   RESTE+2

         LDA   RESTE+2
         BNE   SUITE1
         LDA   RESTE+1
         CMP   LENBUFF
         BCS   SUITE1
         STA   PARMLIT+5
         LDA   RESTE
         STA   PARMLIT+4
         JMP   SUITE2

SUITE1   LDA   £0
         LDY   LENBUFF
         STA   PARMLIT+4
         STY   PARMLIT+5

SUITE2   JSR   LIT

         LDX   PARMLIT+6
         LDY   PARMLIT+7
         TXA
         CLC
         ADC   POSEOF
         STA   POSEOF
         TYA
         ADC   POSEOF+1
         STA   POSEOF+1
         LDA   £0
         ADC   POSEOF+2
         STA   POSEOF+2
         STX   PARMECR+4
         STY   PARMECR+5
         JSR   ECRIT

         LDA   POSEOF+2
         CMP   PARMEOF+4
         BCC   SUITE3
         LDA   POSEOF+1
         CMP   PARMEOF+3
         BCC   SUITE3
         LDA   POSEOF
         CMP   PARMEOF+2
         BEQ   END
SUITE3   JMP   SUITE

END      JSR   CLOT

         LDA   VPATH2
         LDY   VPATH2+1
         STA   PARMSFI+1
         STY   PARMSFI+2
         LDX   £4
RECOP    LDA   PARMGFI+2,X
         STA   PARMSFI+2,X
         DEX
         BNE   RECOP

         JSR   FIXFILE

         RTS

* Ferme le fichier source
 puis le fichier destination

CLOT     LDA   PARMLIT+1
         STA   PARMCLO+1

         JSR   FERME

         LDA   PARMECR+1
         STA   PARMCLO+1

         JSR   FERME

RESTORE  LDA   OLDHIM
         LDY   OLDHIM+1
         STA   HIMEM
         STY   HIMEM+1

         LDA   OLDBUFF
         LDY   OLDBUFF+1
         STA   BUFFER
         STY   BUFFER+1

         RTS

* Met le préfixe du lecteur
 par défaut devant le pathname
* pointé par BUFFER

INCPATH  LDA   £1
         JSR   GBUFF

         LDY   £0
         STA   PARMON+3
         STY   PARMON+2
         STA   PARMPREF+2
         STY   PARMPREF+1
         STA   LITPATH+2
         STY   LITPATH+1

         JSR   PRNDPREF

         JSR   LITPATH
         BEQ   PASPREF
         RTS

PASPREF  LDA   DEFSLT
         ASL
         ASL
         ASL
         ASL
         ASL
         TAX
         LDY   DEFDRV
         DEY
         TYA
         LSR
         TXA
         ROR
         STA   PARMON+1

         JSR   ENLIGNE

         LDA   BUFFER
         CLC
         ADC   £1
         STA   A1
         LDA   BUFFER+1
         ADC   £0
         STA   A1+1
         LDY   £0
         LDA   (BUFFER),Y
         SEC
         ADC   BUFFER
         STA   A2
         LDA   BUFFER+1
         ADC   £0
         STA   A2+1
         JSR   LITPATH
         CLC
         ADC   £3
         CLC
         ADC   BUFFER
         STA   A4
         LDA   BUFFER+1
         ADC   £0
         STA   A4+1

         LDA   A2
         SEC
         SBC   A1
         TAY
         INY
DEPLACE  DEY
         LDA   (A1),Y
         STA   (A4),Y
         CPY   £0
         BNE   DEPLACE

         LDA   PARMON+2
         CLC
         ADC   £1
         STA   A1
         LDA   PARMON+3
         ADC   £0
         STA   A1+1
         JSR   LITPATH
         SEC
         ADC   PARMON+2
         STA   A2
         LDA   PARMON+3
         ADC   £0
         STA   A2+1
         LDA   BUFFER
         CLC
         ADC   £2
         STA   A4
         LDA   BUFFER+1
         ADC   £0
         STA   A4+1

         JSR   MOVE

         LDY   £0
         LDA   (BUFFER),Y
         STA   ADD+1
         JSR   LITPATH
         CLC
ADD      ADC   £0
         ADC   £2
         STA   (BUFFER),Y
         INY
         LDA   £"/"
         STA   (BUFFER),Y
         JSR   LITPATH
         CLC
         ADC   £2
         TAY
         LDA   £"/"
         STA   (BUFFER),Y

LITPATH  LDA   $FFFF
         AND   £%00001111
         RTS

**********
*
* Appels au ProDOS
*
```

```
**********

GETFINFO JSR     MLI
         DFB     GET_FILE_INF●
         DA      PARMGFI
         BCS     ERROR
         RTS
HEUDAT   JSR     MLI
         DFB     GET_TIME
         DFB     ●,0
         BCS     ERROR
         RTS

CREE     JSR     MLI
         DFB     CREATE
         DA      PARMCRE
         BCS     ERROR
         RTS

OUVRE    JSR     MLI
         DFB     OPEN
         DA      PARMOUV
         BCS     ERROR
         RTS

ECRIT    JSR     MLI
         DFB     WRITE
         DA      PARMECR
         BCS     ERROR
         RTS

FERME    JSR     MLI
         DFB     CLOSE
         DA      PARMCLO
         BCS     ERROR
         RTS

GEOF     JSR     MLI
         DFB     GET_EOF
         DA      PARMEOF
         BCS     ERROR
         RTS

LIT      JSR     MLI
         DFB     READ
         DA      PARMLIT
         BCS     ERROR
         RTS

PRNDPREF JSR     MLI
         DFB     GET_PREFIX
         DA      PARMPREF
         BCS     ERROR
         RTS

ENLIGNE  JSR     MLI
         DFB     ONLINE
         DA      PARMON
         BCS     ERROR
         RTS

EFFACE   JSR     MLI
         DFB     DESTROY
         DA      PARMDES
         BCS     ERROR
         RTS

FIXFILE  JSR     MLI
         DFB     SET_FILE_INFO
         DA      PARMSFI
         BCS     ERROR
         RTS

GBUFF    JSR     GETBUFR
         BCS     ERROR
         RTS

* Affichage du code et du
 message d'erreur

ERROR    PHA
         CMP     £$48
         BNE     ERR1
         LDA     PARMCRE+1
         LDY     PARMCRE+2
         STA     PARMDES+1
         STY     PARMDES+2
         JSR     CLOT
         JSR     EFFACE
ERR1     LDA     CODEMSG
         LDA     ERR1+1
         LDY     ERR1+2
         JSR     STROUT
         PLA
         PHA
         JSR     PRBYTE
         LDA     £$A0
         JSR     COUT

         LDA     £":"
         JSR     COUT
         LDA     £$A0
         JSR     COUT

         PLA
         LDX     £28
GETMLI   CMP     MLICODE,X
         BEQ     OUIOUI
         DEX
         BNE     GETMLI
OUIOUI   TXA
         STA     ERR2+1
         ASL
         CLC
ERR2     ADC     £0
         TAX
         INX
         LDA     MLIERR,X
         INX
         LDY     MLIERR,X
         JSR     STROUT

         LDA     £7
         JSR     COUT
         LDA     £$8D
         JSR     COUT
         JSR     RESTORE
         JMP     WARM

* Table des adreeses
 des messages d'erreur

MLIERR   LDA     MLI1
         LDA     MLI2
         LDA     MLI3
         LDA     MLI4
         LDA     MLI5
         LDA     MLI6
         LDA     MLI7
         LDA     MLI8
         LDA     MLI9
         LDA     MLIA
         LDA     MLIB
         LDA     MLIC
         LDA     MLID
         LDA     MLIZ
         LDA     MLIE
         LDA     MLIF
         LDA     MLIG
         LDA     MLIH
         LDA     MLII
         LDA     MLIJ
         LDA     MLIK
         LDA     MLIL
         LDA     MLIM
         LDA     MLIN
         LDA     MLIO
         LDA     MLIP
         LDA     MLIQ
         LDA     MLIR
         LDA     MLIS

         BRK

POSEOF   DS      3
RESTE    DS      3
OLDHIM   DS      2
CALCUL   DS      2
OLDBUFF  DS      2
LENBUFF  DS      1
ECCO     DS      1

COMMAND  STR     'COPY'


PARMGFI  DFB     $0A
         DS      17
PARMSFI  DFB     $07
         DS      13
PARMCRE  DFB     $07
         DS      11
PARMECR  DFB     $04
         DS      7
PARMLIT  DFB     $04
         DS      7
PARMOUV  DFB     $03
         DS      4
REFNUM   DS      1
PARMEOF  DFB     $02
         DS      4
PARMON   DFB     $02
         DS      3


PARMPREF DFB     $01
         DS      2
PARMDES  DFB     $01
         DS      2
PARMCLO  DFB     $01
         DS      1

CODEMSG  HEX     8D
         ASC     " Code $"
         HEX     00

* Codes d'erreur du MLI

MLICODE  HEX     01042527282B2E404243444546474840
                 94A4B4C4D4E5051525355556575A
MLI1     ASC     "Mauvais code de fonction du ML
                 I."
         HEX     00
MLI2     ASC     "Nombre de paramètres incorrect
                 ."
         HEX     00
MLI3     ASC     "Table de vecteurs d'interrupti
                 on pleine."
         HEX     00
MLI4     ASC     "Erreur d'entrée/sortie."
         HEX     00
MLI5     ASC     "Pas de périphérique connecté."
         HEX     00
MLI6     ASC     "Disque protégé à l'écriture."
         HEX     00
MLI7     ASC     "La disquette a été changée."
         HEX     00
MLI8     ASC     "La syntaxe du nom est incorrec
                 te."
         HEX     00
MLI9     ASC     "Il y a déja huit fichiers ouve
                 rts."
         HEX     00
MLIA     ASC     "Ce numéro de référence n'est p
                 as celui d'un"
         ASC     " fichier ouvert."
         HEX     00
MLIB     ASC     "Sous-catalogue inexistant."
         HEX     00
MLIC     ASC     "Le volume n'est pas disponible
                 ."
         HEX     00
MLID     ASC     "Le fichier n'est pas dans le c
                 atalogue demandé."
         HEX     00
MLIZ     ASC     "Le fichier existe déjà."
         HEX     00
MLIE     ASC     "Pas assez de place sur ce volu
                 me."
         HEX     00
MLIF     ASC     "Le catalogue principal est ple
                 in."
         HEX     00
MLIG     ASC     "Le MLI est trop ancien pour li
                 re ce fichier."
         HEX     00
MLIH     ASC     "Ce ProDOS ne peut utiliser ce
                 type de stockage."
         HEX     00
MLII     ASC     "La fin du fichier a été attein
                 te."
         HEX     00
MLIJ     ASC     "On ne peut aller au delà de la
                 fin du fichier."
         HEX     00
MLIK     ASC     "Le fichier est verrouillé."
         HEX     00
MLIL     ASC     "Le fichier est resté ouvert."
         HEX     00
MLIM     ASC     "Le catalogue est endommagé."
         HEX     00
MLIN     ASC     "Le disque n'est pas en ProDOS.
                 "
         HEX     00
MLIO     ASC     "Un des paramètres de la table
                 est incorrect."
         HEX     00
MLIP     ASC     "ProDOS ne peut gerer que huit
                 lecteurs à la fois."
         HEX     00
MLIQ     ASC     "Le buffer ne peut s'implanter
                 à l'endroit désigné."
         HEX     00
MLIR     ASC     "Il y a deux noms de volume ide
                 ntiques."
         HEX     00
MLIS     ASC     "La carte des blocs est endomma
                 gée."
         HEX     00
PFIN
```

# COPIE.TF :

# Copier TOUT les fichiers

Christian Piard

L'objet du présent programme est de répondre aux besoins de sauvegarde de volumes de tailles différentes. Avec MouseDesk, la copie de multiples fichiers ne pose pas de problème mais avec le Filer de ProDOS, pas question de sauvegarder des dossiers ni de copier des volumes différents.

La configuration était la suivante : Apple //, une carte Speedisk 1 Méga, un lecteur 800 Ko, pas de souris et le problème était de saugarder la carte RAM sur une disquette 800 Ko. Le programme Basic présenté ici n'est qu'une utilisation du Copieur de F. Rosay listé dans ces pages : il se charge de rechercher dans le catalogue les dossiers et sous-dossiers quelle qu'en soit la profondeur puis fait appel à la commande externe COPY pour en assurer la sauvegarde.

Il donne accessoirement une façon de rechercher les fichiers dans les dossiers.

## Utilisation

Le programme se lance simplement par :

*RUN COPIE.TF* ou *- COPIE.TF*

exclusivement sous ProDOS.

Il demande successivement les port et lecteur du volume original et du volume destination puis recherche les catalogues, les crée si nécessaire sur le volume destination et enfin exécute les copies.

Si un fichier existe déjà sur la copie, le programme affiche un message auquel on répond par :

- *O* le fichier sera effacé et remplacé,
- *N* la copie de ce fichier est annulée,
- *S* ce fichier et tous les éventuels suivants seront effacés et remplacés.

Pour une question de rapidité, il est vivement conseillé de reformater le volume destination avant copie...

## Remarques

Ce petit programme pourrait être adapté pour faire des sauvegardes d'un volume 800 Ko sur plusieurs 140 Ko par exemple (ici, le cycle de copie s'interrompt en ligne 4040 en cas de saturation du disque destination).

La commande externe COPY gère elle-même les erreurs et ne permet pas de les récupérer par l'ONERR. Les POKEs de la ligne 0 ont pour objet de détourner les erreurs vers le vecteur $BE09-$BE0B (Errout) de la page globale du Basic.System ; ainsi un 'Duplicate File Name' est bien pris en charge par l'ONERR GOTO 4000.

La ligne 100 prévoit 200 dossiers et sous-dossiers et 500 fichiers par dossier au maximum : la modification des DIM devrait répondre à tous les besoins.

La commande COPY ne gère pas les numéros de port et de lecteur aussi, il convient de nommer différemment les volumes origine et destination.



## Programme COPIE.TF

```
0  PRINT  CHR$ (4)"BLOADCOPY": POKE 2578
    3,76: POKE 25784,3: POKE 25785,101:
    POKE 25860,9: POKE 25861,190: CALL 2
    4576
1  HOME : FOR I = 0 TO 9: READ Z: POKE 7
    68 + I,Z: NEXT : DATA 104,168,104,16
    6,223,154, 72,152,72,96
2  ONERR  GOTO 4000
5 D$ =  CHR$ (4)
20  PRINT D$"PREFIX": INPUT AP$
100 AN = 0:NN = 0:NK = 0:DF = 0: DIM DI$
    (200),FI$(500)
105  PRINT "Origine": PRINT "-------"
110  VTAB 5: INPUT "Port : ";S$: ON S$ <
     "1" OR S$ > "7" GOTO 110:S =  VAL (
    S$)
111  VTAB 6: INPUT "Lect : ";U$: ON U$ <
     "1" OR D$ > "2" GOTO 111:D =  VAL (
    U$)
112  PRINT D$"PREFIX,S"S",D"D: PRINT D$"
    PREFIX": INPUT OG$:OG$ =  LEFT$ (OG$
    , LEN (OG$) - 1)
115  PRINT : PRINT : PRINT "Destination"
    : PRINT "-----------"
```

# Exemple d'exécution

//e
//e+
//c
][gs

ProDOS

```
Origine
-------
Port :  1
Lect :  1

Destination
-----------
Port :  5
Lect :  2

Copie de /SPDK vers /COPIE.SPDK


Ok ?  O


Recherche des catalogues...


/SPDK
/SPDK/MERLIN
/SPDK/MERLIN/LIB
/SPDK/MERLIN/UTIL

Création des catalogues...

/COPIE.SPDK/MERLIN
/COPIE.SPDK/MERLIN/LIB
/COPIE.SPDK/MERLIN/UTIL

Copie des fichiers


Copie de : /SPDK/PRODOS
-------> : /COPIE.SPDK/PRODOS

Copie de : /SPDK/REBOOT.SYSTEM
-------> : /COPIE.SPDK/REBOOT.SYSTEM

Copie de : /SPDK/BASIC.SYSTEM
-------> : /COPIE.SPDK/BASIC.SYSTEM
---------------------------------------

Copie de : /SPDK/MERLIN/MERLIN.SYSTEM
-------> : /COPIE.SPDK/MERLIN/MERLIN.SYSTEM
---------------------------------------

Copie de : /SPDK/MERLIN/LIB/SENDMSG.S
-------> : /COPIE.SPDK/MERLIN/LIB/SENDMSG.S
---------------------------------------

Copie de : /SPDK/MERLIN/UTIL/XREF
-------> : /COPIE.SPDK/MERLIN/UTIL/XREF

Copie de : /SPDK/MERLIN/UTIL/XREFA
-------> : /COPIE.SPDK/MERLIN/UTIL/XREFA
---------------------------------------
```

```
120  VTAB 13: INPUT "Port : ";S$: ON S$
    < "1" OR S$ > "7" GOTO 120:S =  VAL
    (S$)
121  VTAB 14: INPUT "Lect : ";U$: ON U$
    < "1" OR D$ > "2" GOTO 121:D =  VAL
    (U$)
122  PRINT D$"PREFIX,S"S",D"D: PRINT D$"
    PREFIX": INPUT DE$:DE$ =  LEFT$ (DE$
    , LEN (DE$) - 1)
125  IF OG$ = DE$ THEN  PRINT : PRINT "L
    es volumes sont homonymes...": WAIT
    49152,128: POKE 49168,0: PRINT D$"PR
    EFIX"AP$: RUN 1
130  PRINT : PRINT "Copie de "OG$" vers
    "DE$
140  PRINT : INPUT "Ok ? ";R$: IF R$ <
    > "O" AND R$ <  > "o" THEN  RUN 1
150  PRINT : PRINT
155  REM ***************
156  REM * Recherche des catalogues
157  REM ***************
160  PRINT "Recherche des catalogues..."
    : PRINT
190 DI$(0) = OG$
200 DT = 0
210  FOR I = AN TO NN
220 PA$ = DI$(I)
221  PRINT PA$
222  PRINT D$"OPEN"PA$",TDIR
224  PRINT D$"READ"PA$
226 F = 0
228  INPUT A$
230  IF F AND  NOT  LEN (A$) THEN 238
232  IF  NOT  LEN (A$) THEN F = 1
234  IF  MID$ (A$,18,3) = "DIR" THEN DT
    = DT + 1:DI$(NN + DT) = PA$ + "/" +
     MID$ (A$,2,15): GOSUB 3000
236  GOTO 228
238  PRINT D$"CLOSE
240  NEXT
250  IF DT THEN AN = NN + 1:NN = NN + DT
    :DT = 0: GOTO 210
260 NC = NN + DT
270  REM ***************
271  REM * Création des catalogues
272  REM ***************
280  PRINT : PRINT "Création des catalog
    ues...": PRINT
290  IF  NOT NC GOTO 400
300  FOR I = 1 TO NC:PA$ = DI$(I): FOR J
     = 2 TO  LEN (PA$): IF  MID$ (PA$,J,
    1) = "/" THEN PA$ =  MID$ (PA$,J + 1
    ):J =  LEN (PA$)
310  NEXT
315  PRINT DE$"/"PA$
320  PRINT D$"CREATE"DE$"/"PA$",TDIR
330  NEXT
335  REM ***************
336  REM * Copie des fichiers
337  REM ***************
```

```
340  PRINT : PRINT "Copie des fichiers":
      PRINT
400  FOR I = 0 TO NC
405 N1 = 0
410  PRINT D$"OPEN"DI$(I)",TDIR": PRINT
    D$"READ"DI$(I)
420 F = 0
425  INPUT A$: INPUT A$
430  REM
431  INPUT A$: IF F AND  NOT  LEN (A$) T
    HEN 500
440  IF  NOT  LEN (A$) THEN F = 1: GOTO
    430
450  IF  MID$ (A$,18,3) = "DIR" THEN 430
455 F1$ = DI$(I) + "/" +  MID$ (A$,2,15)
460 N1 = N1 + 1:FI$(N1) = F1$
470  GOTO 430
500  PRINT D$"CLOSE
501  IF  NOT N1 THEN 510
502  FOR K = 1 TO N1:F1$ = FI$(K):F2$ =
    DE$ +  RIGHT$ (F1$, LEN (F1$) -  LEN
     (OG$))
503  PRINT : PRINT "Copie de : "F1$
504  PRINT D$"COPY"F1$","F2$: PRINT "---
    ----> : "F2$
505  NEXT K
509  FOR K = 0 TO 38: PRINT "-";: NEXT :
      PRINT
510  NEXT I
997  PRINT D$"PREFIX"AP$
999  END
1999  REM ***************
3000 U$ = DI$(NN + DT):U =  LEN (U$): FO
     R J = 1 TO U: IF  MID$ (U$,J,1) = "
     " THEN DI$(NN + DT) =  LEFT$ (U$,J -
      1):J = U
3010  NEXT : RETURN
4000  CALL 768: REM Bug de l'OnErr
4010 LI =  PEEK (218) + 256 *  PEEK (219
     ):ER =  PEEK (222)
4020  IF ER = 19 AND LI = 320 THEN  GOTO
      330
4030  IF  NOT (ER = 71 AND LI = 504) THE
     N 4035
4031  ON DF GOTO 4034: PRINT  SPC( 11)F2
     $" existe déjà": PRINT "Remplcmt : O
     /N    Remplacer tous : S
4032  WAIT 49152,128: POKE 49168,0:RR =
      PEEK (49152): IF RR =  ASC ("N") OR
      RR =  ASC ("n") THEN  PRINT "Copie
     annulée": PRINT : GOTO 505
4033  IF RR =  ASC ("S") OR RR =  ASC ("
     s") THEN DF = 1
4034  ON RR <  >  ASC ("O") AND RR <  >
      ASC ("o") AND  NOT DF GOTO 4032: PR
     INT D$"UNLOCK"F2$: PRINT D$"DELETE"F
     2$: PRINT "Effacemt : "F2$: RESUME
4035  PRINT : PRINT  CHR$ (7)
4040  IF  NOT ER THEN  PRINT "Disque des
     tination plein": END
4099  PRINT D$"PREFIX"AP$
4100  PRINT "Erreur n° "ER" ligne "LI
```

# Pascal :

# Fichiers séquentiels indexés

Jean-Louis Chauvin

Avec un Apple //, il est possible de gérer des gros fichiers de données et, pour que le temps d'accès aux articles reste acceptable, il est d'usage d'organiser le fichier en fichier 'séquentiel indexé', comportant une table d'index associant à la "clé" de chaque article son numéro dans le fichier.

À partir de quelques centaines d'articles, la table ne tiendrait plus entière en mémoire et serait, sur disque, trop longue à explorer. Il faut alors la remplacer par un système de tables multiples, dont la gestion est à inclure dans le programme.

Nous vous proposons ici une "UNIT" assurant la gestion d'un système de tables à deux niveaux.

### Les programmes

***Vous disposez de la disquette d'accompagnement 800 Ko:***

*Les fichiers ont été transférés en ProDOS grâce à Universal File Conversion (UFC). C'est ce programme que vous utiliserez pour les repasser sous Pascal.*

***Vous disposez de la disquette d'accompagnement 140 Ko :***

*Deux possibilités :*

*Sur la face ProDOS, il vous faudra utiliser UFC comme sur la disquette 800Ko.*
*Les fichiers sont également en format DOS 3.3 sur l'autre face : utiliser le programme BASIC-PASCAL livré sur la disquette pour repasser les fichiers sous Pascal.*

Vous pourrez l'utiliser, grâce au linker, avec le programme de gestion de fichier que vous écrirez selon vos besoins, en partant du modèle présenté. Les performances sont les suivantes : pour un fichier de 450 articles, l'extraction d'un d'entre eux par sa clé demande environ une seconde. Le tri alphabétique demande aussi environ une seconde par article. En outre, le fichier n'a pas à être purgé pour récupérer la place des articles supprimés.

## La démonstration

Le programme 'REPERTOIRE' gère un fichier d'adresses d'une capacité de 450 articles. La clé de chaque article comprend le nom et le prénom concaténés, puis tronqués à 15 caractères.

Vous entrez d'abord le nom du fichier, par exemple "ADRESSES". En cas d'absence sur le disque, il y a création du fichier de données ADRESSES.DATA et du fichier de tables ADRESSES.TABL (attention à la place disponible). Vous pouvez ensuite ajouter des articles, les rechercher un par un par le début de leur clé (pour éventuellement les modifier ou les supprimer), ou encore les lister par ordre alphabétique de clé à partir d'une origine quelconque.

## Le fonctionnement

Les fonctions principales du programme P.REPERT sont assurées par les procédures AJOUTER, RECHERCHER et LISTER. Celles-ci font appel à des procédures de l'unité link U.GESTABL :

- CLASSER range dans les tables la clé et le numéro d'un article ;
- DECLASSER supprime une clé des tables et les réorganise ;
- REPERER et DECOUVRIR cherchent dans les tables le numéro d'article associé à une clé donnée (respectivement dans l'ordre descendant ou ascendant) ;
- LOCALISER cherche la position d'une clé dans les tables.

### Structure des tables

Les tables sont composées d'éléments associant une clé et un index. Chaque table a une capacité de *DIM* éléments (plus un élément de rang 0 qui contient le nombre d'éléments présents). Le fichier comprend une table primaire, de numéro 0, et *DIM* tables secondaires. Les tables secondaires comprennent les clés de tous les articles, associées aux numéros d'article.

### *Exemple de tables pour DIM=4*

```
Fichier DONNEES          Fichier INDEX

0   | 6|                           0        1        2        3        4
1   |ANDRE,....|          0   |___,3  |ALA,1  |JEA,3  |REN,2  |___,__|
2   |JULES,....|          1   |___,3  |ALA,4  |AND,1  |DAV,3  |___,__|
3   |DAVID,....|          2   |___,1  |REN,5  |___,__|___,__|___,__|
4   |ALAIN,....|         *3   |___,2  |JEA,6  |JUL,2  |___,__|___,__|
5   |RENE ,....|          4   |___,0  |___,__|___,__|___,__|___,__|
6   |JEAN ,....|
```

La table primaire contient les clés de tête des tables secondaires, associées aux numéros de table.

### Procédures de l'UNIT

Chacune des cinq procédures cherche d'abord dans la table 0 le numéro de la table secondaire concernée. Dans l'exemple repéré en *italique*, pour DECOUVRIR, le numéro d'article défini par la clé JUL, la table 0 aiguille sur la table 3 (car JEA<=JUL<REN), et celle-ci indique le numéro 2 de l'article.

La procédure CLASSER fonctionne de façon à maintenir vide la dernière position de chaque table secondaire. Si, après insertion de l'élément (clé, numéro), cette table contient *DIM* éléments, elle est dégonflée par transfert de son dernier élément :

- soit dans la table de rang suivant si celle-ci contient moins de *DIM-1* éléments ;
- sinon dans une nouvelle table, dont la clé est insérée à son tour dans la table 0.

### Procédures du programme

Elles fonctionnent comme suit :

- MODIFIER : si la nouvelle clé diffère de l'ancienne, celle-ci est déclassée, puis la nouvelle est classée ;
- SUPPRIMER : l'ancienne clé est déclassée, puis une clé vide (chaîne '') est classée, associée au même numéro d'article. Les articles supprimés peuvent être repérés par leurs clés vides en tête des tables ;
- AJOUTER : si les tables comportent des clés vides en tête, le dernier des articles vides est extrait, puis modifié par substitution des nouvelles données à l'ancien contenu vide. Sinon, un nouvel article est créé et sa clé est classée dans les tables.

Ainsi, les emplacements des articles supprimés sont réutilisés automatiquement, et le fichier n'a pas à être purgé.

### Saturation

Au remplissage, les tables se saturent, dans le cas le moins favorable, avec alternativement une table à 1 élément et une table à *DIM-1* éléments. Pour éviter la saturation, le nombre d'articles est plafonné à *DIM*(DIM DIV 2)*. Ainsi, dans P.REPERT, DIM=30 et le plafond est à 450 articles.

Mais, en supprimant ou en modifiant des articles, certaines tables peuvent se dégarnir. Pour éviter une saturation prématurée, les procédures suivantes interviennent si nécessaire :

- TASSER qui réunit deux tables de rang consécutif dont l'effectif total est inférieur à *DIM* ;
- COMPRIMER renumérote les tables en supprimant les tables vides.

## Création d'un nouveau programme

Ajustez d'abord, si nécessaire et en tête de U.GESTABL.TEXT, la valeur de *DIM* d'après la taille de votre fichier de données, ainsi que le nombre de caractères de la clé, puis compilez cette unité.

Écrivez ensuite votre programme de gestion de fichier, de préférence à partir de P.REPERT.TEXT, en évitant de modifier la mise en œuvre des procédures de gestion de tables dans AJOUTER, RECHERCHER et LISTER. Après l'avoir compilé, il ne vous reste plus qu'à assembler avec le linker le code de l'unité dans celui du programme, puis à exécuter le code résultant.

## Programme 'U.GESTABL'

*Note : le caractère '∫' indique la continuité de la ligne*

```
(*$S+*)
UNIT Sequin_Gestables;

INTERFACE
  CONST Dim=30; (*a ajuster selon taille fichier*)
  TYPE T_cle=STRING[15] (*a ajuster*);
    T_elem=RECORD
              Cle:T_cle;
              Index:INTEGER;
           END;
    T_table=ARRAY[0..Dim] OF T_elem;
  VAR Fitab:FILE OF T_table;
    Numero,Num_2,Rang:INTEGER;
    Table1,Table2:T_table;
  PROCEDURE Classer(Libelle:T_cle;Numero:INTEGER);
  PROCEDURE Declasser(Libelle:T_cle);
  PROCEDURE Reperer(Donnee:T_cle);
  PROCEDURE Decouvrir(Donnee:T_cle);
  PROCEDURE Localiser(Donnee:T_cle;VAR Rang1,Rang2:
INTEGER);

IMPLEMENTATION

  PROCEDURE Situer(Libelle:T_cle; Table:T_table; V∫
 AR Rang:INTEGER);
    BEGIN
      Rang:=1;
      WHILE (Libelle>=Table[Rang].Cle) AND (Rang<T∫
     able[0].Index)
        DO Rang:=Rang+1;
      IF (Libelle>=Table[Rang].Cle) AND (Rang=Tabl∫
     e[0].Index)
        THEN Rang:=Rang+1;
      Rang:=Rang-1;
    END;

  PROCEDURE Inserer(Element:T_elem; VAR Table:T_ta∫
 ble);
    VAR Indice, Rang:INTEGER;
```

```
    BEGIN
      Situer(Element.Cle,Table,Rang);
      FOR Indice:=Table[0].Index DOWNTO (Rang+1) DO
Table[Indice+1]:=Table[Indice];
      Table[Rang+1]:=Element;
      Table[0].Index:=Table[0].Index+1;
    END;

  PROCEDURE Comprimer;
    (*Supprime les tables vides*)
    VAR I,Nombre,Trou,Rang:INTEGER;
    BEGIN
      Trou:=0;Nombre:=Table1[0].Index;
      FOR I:=1 TO Nombre DO
        IF Table1[I].Cle='?' THEN Trou:=Trou+1
          ELSE Table1[I-Trou]:=Table1[I];
      Table1[0].Index:=Table1[0].Index-Trou;
      Trou:=0;
      FOR I:=1 TO Nombre DO BEGIN
        SEEK(Fitab,I);GET(Fitab);
        IF Fitab^[0].Index=0 THEN Trou:=Trou+1
          ELSE BEGIN
            SEEK(Fitab,I-Trou);PUT(Fitab);
            Rang:=1;
            WHILE (Table1[Rang].Index<>I) AND (Ran∫
            g<Nombre)
               DO Rang:=Rang+1;
            Table1[Rang].Index:=I-Trou;
            Table1[Rang].Cle:=Fitab^[1].Cle;
          END;
      END;
    END;

  PROCEDURE Tasser;
    (*Reunit 2 Tables peu pleines et comprime*)
    VAR I,Indice,Nombre,Num_A,Num_B:INTEGER;
      TableA,TableB:Ttable;
    BEGIN
      I:=1;
      WHILE I<Table1[0].Index DO
      BEGIN
        Num_A:=Table1[I].Index;
        SEEK(Fitab,Num_A);GET(Fitab);TableA:=Fitab∫
        ^;
        Num_B:=Table1[I+1].Index;
              SEEK(Fitab,Num_B);GET(Fitab);TableB∫
             :=Fitab^;
        Nombre:=TableA[0].Index+TableB[0].Index;
        IF Nombre<Dim THEN
        BEGIN
          FOR Indice:=1 TO TableB[0].Index DO
            TableA[TableA[0].Index+Indice]:=TableB∫
            [Indice];
          TableB[0].Index:=0;TableA[0].Index:=Nomb∫
         re;
          Table1[I+1].Cle:='?';
          SEEK(Fitab,Num_A);Fitab^:=TableA;PUT(Fit∫
         ab);
          SEEK(Fitab,Num_B);Fitab^:=TableB;PUT(Fit∫
         ab);
          I:=I+1;
        END;
        I:=I+1;
      END;
      Comprimer;
    END;

  PROCEDURE Classer(*Libelle:T_cle;Numero:INTEGER*∫
 );
    (*Insere dans les tables la cle et le numero d∫
   'un article*)
    VAR Rang, Num_2:INTEGER;Element:T_elem;


    PROCEDURE Degonfler( Rang_A:INTEGER;VAR Table0∫
   , TableA:T_table);
      (*Reporte dans une autre table le dernier el∫
     em. d'une table pleine*)
      VAR TableB:T_table; Element:T_elem;
        Rang_B, Num_B:INTEGER;
      BEGIN
        Rang_B:=Rang_A+1;
        Num_B:=Table0[Rang_B].Index;
        SEEK(Fitab,Num_B); GET(Fitab);
        TableB:=Fitab^;
        Inserer(TableA[Dim],TableB);
        TableA[0].Index:=Dim-1;
        IF (TableB[0].Index=Dim) OR (Rang_A=Table0∫
       [0].Index)
                          (*TableB pleine ou Table∫
                         A en queue dans Table0*)
          THEN
            BEGIN
              Num_B:=Table0[0].Index+1;
              TableB[0].Index:=0;
              Inserer(TableA[Dim],TableB);
              Element.Cle:=TableB[1].Cle;
              Element.Index:=Num_B;
              Inserer(Element,Table0);
            END
          ELSE
            Table0[Rang_B].Cle:=TableB[1].Cle;
        SEEK(Fitab,Num_B); Fitab^:=TableB;
        PUT(Fitab);
      END;

    BEGIN (*Classer*)
      Situer(Libelle,Table1,Rang);
      IF Rang=0 THEN Rang:=1;
      Num_2:=Table1[Rang].Index;
      SEEK(Fitab,Num_2); GET(Fitab);
      Table2:=Fitab^;
      Element.Cle:=Libelle;
      Element.Index:=Numero;
      Inserer(Element,Table2);
      IF Table1[0].Index=0 THEN Table1[0].Index:=1;
      Table1[Rang].Cle:=Table2[1].Cle;
      IF Table2[0].Index=Dim THEN Degonfler(Rang,T∫
     able1,Table2);
      SEEK(Fitab,Num_2); Fitab^:=Table2; PUT(Fitab∫
     );
      IF Table1[0].Index=Dim THEN Tasser;
      SEEK(Fitab,0); Fitab^:=Table1; PUT(Fitab);
    END;

  PROCEDURE Declasser(*Libelle:T_cle*);
    (*Retire un element de la Table2*)
    VAR Indice,Rang2:INTEGER;
    BEGIN
      Rang2:=Table2[0].Index;
              WHILE Libelle<>Table2[Rang2].Cl∫
             e DO Rang2:=Rang2-1;
      Table2[0].Index:=Table2[0].Index-1;
      FOR Indice:=Rang2 TO Table2[0].Index DO
        Table2[Indice]:=Table2[Indice+1];
      Table1[Rang].Cle:=Table2[1].Cle;
      Fitab^:=Table2;
      SEEK(Fitab,Num_2); PUT(Fitab);
```

][+
//e
//e+
//c
][gs

```
    IF Table2[0].Index=0 THEN BEGIN
      Table1[Rang].Cle:='?';
      Comprimer;END;
  END;

 PROCEDURE Reperer(*Donnee:T_cle*);
   (*Repere un article d'apres sa cle*)

  VAR Rang2:INTEGER;
  BEGIN
    Situer(Donnee,Table1,Rang);
    Num_2:=Table1[Rang].Index;
    SEEK(Fitab,Num_2); GET(Fitab);
    Table2:=Fitab^;
    Rang2:=Fitab^[0].Index;
    WHILE (Donnee<>Fitab^[Rang2].Cle) AND (Rang2∫
   >0)
      DO Rang2:=Rang2-1;
    Numero:=Fitab^[Rang2].Index;
    IF Rang2=0 THEN Numero:=0;
  END;


 PROCEDURE Decouvrir(*Donnee:T_cle*);
  VAR Rang2:INTEGER;
  BEGIN
    Situer(Donnee,Table1,Rang);
    IF (Rang<Table1[0].Index) AND (POS(Donnee,Ta∫
   ble1[Rang+1].Cle)=1)
      THEN Rang:=Rang+1;
    IF Rang=0 THEN Rang:=1;
    Num_2:=Table1[Rang].Index;
    SEEK(Fitab,Num_2);GET(Fitab);
    Table2:=Fitab^;
            Rang2:=1;
    WHILE (POS(Donnee,Fitab^[Rang2].Cle)<>1) AND
      (Rang2<Fitab^[0].Index) DO Rang2:=Rang2+1;
    Numero:=Fitab^[Rang2].Index;
    IF POS(Donnee,Fitab^[Rang2].Cle)<>1 THEN Num∫
   ero:=0;
  END;

 PROCEDURE Localiser(*Donnee:T_cle;VAR Rang1,Rang∫
2:INTEGER*);
  BEGIN
    Situer(Donnee,Table1,Rang1);IF Rang1=0 THEN ∫
   Rang1:=1;
    Num_2:=Table1[Rang1].Index;
    SEEK(Fitab,Num_2);GET(Fitab);
    Rang2:=1;
    WHILE (Donnee>Fitab^[Rang2].Cle) AND (Rang2<∫
   =Fitab^[0].Index)
      DO Rang2:=Rang2+1;
    IF Rang2>Fitab^[0].Index THEN BEGIN Rang2:=1∫
   ;Rang1:=Rang1+1;END;
  END;

BEGIN
  Numero:=0;Num_2:=0;Rang:=0;
END.
```

## Programme 'P.REPERT'

***Note : le caractère '∫' indique la continuité de la ligne***

```
PROGRAM Repertoire;
(*fichier sequentiel indexe*)
USES APPLESTUFF,
  (*$U #5:U.GESTABL.CODE *) Sequin_gestables;
TYPE String20=STRING[20];
  String5=STRING[5];
VAR Fich:FILE OF RECORD CASE INTEGER OF
           0: (Der_article:INTEGER);
           1: (Nom,Prenom,Adresse1,Adresse2,Ville,∫
           Telephone:String20;
                Code_postal:String5);
          END;
  Fin,Plafond:INTEGER;
  Cle_1:T_cle;
  Choix:CHAR; Erreur:BOOLEAN;

 PROCEDURE Synthese(Nom,Prenom:String20;VAR Raccou∫
 rci:T_cle);
   (*Cree la cle avec nom et prenom*)
   VAR Nom_prenom:STRING[41];
   BEGIN
     Nom_prenom:=CONCAT(Nom,Prenom);
     IF LENGTH(Nom_prenom)>15 THEN Raccourci:=COPY∫
     (Nom_prenom,1,15)
       ELSE Raccourci:=Nom_prenom;
   END;

PROCEDURE Initfichier;
  TYPE T_nom=STRING[23];
  VAR Test:CHAR;
    Nom_fichier,Nom_table:T_nom;

 PROCEDURE Nommer(VAR Fichier,Table:T_nom);
   VAR Nom,Volume:STRING[10];Lecteur:CHAR;
   BEGIN
     WRITE('nom du fichier (max. 10 caracteres):'∫
     );
     READLN(Nom);
     WRITE('lecteur 1 OU 2 ? ');READ(Lecteur);WRI∫
     TELN;
       STR(ORD(Lecteur)-45,Volume);
     Fichier:=CONCAT('#',Volume,':',Nom,'.DATA');
     Table:=CONCAT('#',Volume,':',Nom,'.TABL');
   END;

 PROCEDURE Creer;
   VAR Nombre:INTEGER;Reponse:STRING[5];
   BEGIN
     WRITE('taper CREER pour confirmer: ');
     READLN(Reponse);
     IF Reponse='CREER' THEN
     BEGIN
       REWRITE(Fich,Nom_fichier);
       Fich^.Der_article:=0; PUT(Fich);
       WRITE('reserver combien de fiches ?'); REA∫
       DLN(Nombre);
       Fich^.Der_article:=1;
       FOR Numero:=1 TO Nombre DO PUT(Fich);
       CLOSE(Fich,LOCK);REWRITE(Fitab,Nom_table);
       Fitab^[0].Index:=0;Fitab^[0].Cle:='';
       FOR Numero:=1 TO Dim DO
         BEGIN
           Fitab^[Numero].Index:=1;
           Fitab^[Numero].Cle:=' ';
         END;
       FOR Numero:=0 TO Dim DO PUT(Fitab);
       CLOSE(Fitab,LOCK);
     END;
   END;
```

```
BEGIN (*Initfichier*)
  Erreur:=FALSE;
  REPEAT
    Nommer(Nomfichier,Nomtable);
    (*$I-*)
    CLOSE(Fich);RESET(Fich,Nomfichier);
    (*$I+*)
    IF IORESULT=0 THEN Test:='O'
    ELSE BEGIN
      WRITELN('fichier introuvable - voulez-vous∫
      :');
      WRITELN('- le Creer');
                     WRITELN('- Modifier son nom∫
                     ');
      READ(Test);WRITELN;
      IF Test='C' THEN Creer;
    END;
  UNTIL Test<>'M';
  (*$I-*)
  CLOSE(Fich); RESET(Fich,Nom_fichier);
  (*$I+*)
  IF IORESULT<>0
    THEN BEGIN
      WRITELN('traitement impossible');
      READ(Choix);
      Erreur:=TRUE;END
    ELSE BEGIN
      Fin:=Fich^.Der_article;
      CLOSE(Fitab);RESET(Fitab,Nom_table);
      Table1:=Fitab^;
    END;
END;


PROCEDURE Masque;
  BEGIN
    PAGE(OUTPUT);
    WRITELN('fiche individuelle n[ ',Numero);
    WRITELN('========================');
    GOTOXY(0,4);WRITELN('nom');WRITE('prenom');
    GOTOXY(0,7);WRITE('adresse');
    GOTOXY(0,10);WRITE('code, ville');
    GOTOXY(0,12);WRITELN('telephone');
  END;

PROCEDURE Saisir;
  VAR Reponse:CHAR;Chaine:String20;

  PROCEDURE Lire(X,Y,L:INTEGER;VAR Champ:String20);
    VAR Donnee:STRING[21];
    BEGIN
      GOTOXY(X+L,Y);WRITE('<');
      GOTOXY(X,Y);READLN(Donnee);
      IF LENGTH(Donnee)>L THEN Donnee:=COPY(Donnee∫
      ,1,L);
      IF Donnee<>'' THEN Champ:=Donnee;
                  GOTOXY(X,Y);WRITE(Champ,CHR(29)∫
                  ); (*effacement fin de ligne*)
    END;

  BEGIN (*Saisir*)
    REPEAT
      Lire(14,4,20,Fich^.Nom);
      Lire(14,5,20,Fich^.Prenom);
      Lire(14,7,20,Fich^.Adresse1);
      Lire(14,8,20,Fich^.Adresse2);
      Chaine:=Fich^.Code_postal;
      Lire(14,10,5,Chaine);Fich^.Code_postal:=Chai∫
      ne;
      GOTOXY(20,10);WRITE(Fich^.Ville);
      Lire(20,10,20,Fich^.Ville);
      Lire(14,12,20,Fich^.Telephone);
      GOTOXY(14,16);WRITE('accord (O/N)? ');
      READ(Reponse);WRITELN;
    UNTIL Reponse='O';
  END;

PROCEDURE Afficher;
  BEGIN
    GOTOXY(14,4);WRITE(Fich^.Nom);
    GOTOXY(14,5);WRITE(Fich^.Prenom);
    GOTOXY(14,7);WRITE(Fich^.Adresse1);
    GOTOXY(14,8);WRITE(Fich^.Adresse2);
    GOTOXY(14,10);WRITE(Fich^.Code_postal);
    GOTOXY(20,10);WRITE(Fich^.Ville);
    GOTOXY(14,12);WRITE(Fich^.Telephone);
    WRITELN;
  END;

PROCEDURE Effacer;
  BEGIN
    Fich^.Nom:='';
    Fich^.Prenom:='';
    Fich^.Adresse1:='';
    Fich^.Adresse2:='';
    Fich^.Code_postal:='';
    Fich^.Ville:='';
    Fich^.Telephone:='';
  END;

PROCEDURE Ajouter;
(*Ajoute une nouvelle fiche*)
  VAR Choix1:CHAR;Sature:BOOLEAN;

  PROCEDURE Auto;
    (*Cree une fiche aleatoire, pour essai*)
    VAR Chaine:String20;

    PROCEDURE Alea (Longueur:INTEGER;VAR Chaine:St∫
    ring20);
      (*Cree une chaine aleatoire*)
      VAR I,L:INTEGER;
      BEGIN
        Chaine:='
        L:=RANDOM MOD Longueur+1;
        FOR I:=1 TO L DO Chaine[I]:=CHR(RANDOM MO∫
        D 26+65);
        Chaine:=COPY(Chaine,1,L);
      END;

    BEGIN (*Auto*)
      RANDOMIZE;
      Alea(20,Fich^.Nom);
      Alea(20,Fich^.Prenom);
      Alea(20,Fich^.Adresse1);
      Alea(20,Fich^.Adresse2);
      Alea(20,Fich^.Ville);
      Alea(20,Fich^.Telephone);
      Alea(5,Chaine);Fich^.Code_postal:=Chaine;
      Afficher;
    END;

  BEGIN (*Ajouter*)
    IF (Fin<Plafond) OR (Table1[1].Cle='')THEN Sat∫
    ure:=FALSE
      ELSE Sature:=TRUE;
    Choix1:='O';
    WHILE (Choix1='O') AND (Sature=FALSE) DO
```

```
    BEGIN
      IF Table1[1].Cle='' (*si des fiches ont ete ∫
      effacees*)
        THEN BEGIN Reperer('');Declasser('');END
        ELSE BEGIN Fin:=Fin+1;Numero:=Fin;END;
      Effacer;
      Masque;
            IF Choix='à' THEN Auto (*pour essai*∫
            ) ELSE
        Saisir;
      SEEK(Fich,Numero);PUT(Fich);
      Synthese(Fich^.Nom,Fich^.Prenom,Cle_1);
      SEEK(Fich,0); Fich^.Der_article:=Fin; PUT(Fi∫
      ch);
      Classer(Cle_1,Numero);
      IF (Fin<Plafond) OR (Table1[1].Cle='') THEN ∫
      Sature:=FALSE
        ELSE Sature:=TRUE;
      IF Sature THEN WRITELN('fichier sature')
        ELSE BEGIN
          GOTOXY(14,18);WRITE('fiche suivante (O/N∫
          )?');
          READ(Choix1); WRITELN;
        END;
    END;
  END;

PROCEDURE Rechercher;
  (*Recherche un article d'apres sa cle*)
  VAR Choix_M:CHAR;Nom,Prenom:String20;

  PROCEDURE Modifier;
    (*Modifie un article*)
    VAR Cle_2:T_cle;
    BEGIN
      GOTOXY(14,18);WRITE(CHR(29)); (*efface ligne∫
      *)
      Synthese(Fich^.Nom,Fich^.Prenom,Cle_1);
      Saisir;
      Synthese(Fich^.Nom,Fich^.Prenom,Cle_2);
      IF Cle_2<>Cle_1 THEN BEGIN
        Declasser(Cle_1);
        Classer(Cle_2,Numero);
      END;
      SEEK(Fich,Numero); PUT(Fich);
    END;

  PROCEDURE Supprimer;
    (*Supprime un article*)
    VAR Reponse:CHAR;
    BEGIN
                WRITE('taper X pour confirmer:')∫
                ;READ(Reponse);WRITELN;
      IF Reponse='X' THEN
        BEGIN
          Synthese(Fich^.Nom,Fich^.Prenom,Cle_1);
          Declasser(Cle_1);
          Classer('',Numero);
          Effacer;
          SEEK(Fich,Numero);PUT(Fich);
        END;
    END;

  BEGIN (*Rechercher*)
    Choix:='O';
    WHILE Choix='O' DO
   BEGIN
     PAGE(OUTPUT);ChoixM:='N';
     WRITE('nom ? '); READLN(Nom);
     WRITE('prenom ? ');READLN(Prenom);
     Synthese(Nom,Prenom,Cle_1);
     Decouvrir(Cle_1);
     IF Numero=0 THEN WRITELN('nom inconnu')
       ELSE BEGIN
         SEEK(Fich,Numero); GET(Fich);
         Masque; Afficher;
         GOTOXY(14,18);
         WRITE('voulez-vous Modifier ou Supprime∫
         r cette fiche ?');
         READ(ChoixM);WRITELN;
         IF ChoixM='M' THEN Modifier;
         IF ChoixM='S' THEN Supprimer;
       END;
     GOTOXY(14,20); WRITE('autre fiche (O/N)?');
     READ(Choix);WRITELN;
   END;
  END;

PROCEDURE Lister;
  VAR I1,I2:INTEGER; Choix:CHAR; Debut:Tcle;
  BEGIN
    WRITE('debut de la liste ? ');READLN(Debut);
    IF Debut='' THEN Debut:='!';(*pour eviter de l∫
    ister les fiches effacees*)
          Localiser(Debut,I1,I2);
    PAGE(OUTPUT);
    WRITELN('pour arreter, taper sur une fleche');
    WRITELN('pour quitter, taper Q'); WRITELN;
    WRITE('numero                nom              '∫
    );
    WRITELN('prenom                ville');
    WRITELN;
    Choix:=' ';
    WHILE (I1<=Table1[0].Index) AND (Choix<>'Q') DO
    BEGIN
      Num_2:=Table1[I1].Index;
      SEEK(Fitab,Num_2); GET(Fitab); Table2:=Fitab∫
      ^;
      Numero:=Table2[I2].Index;
      SEEK(Fich,Numero); GET(Fich);
      WRITE(Numero:4,'  ',Fich^.Nom:20,' ');
      WRITELN(Fich^.Prenom:20,' ',Fich^.Ville:20);
      IF KEYPRESS THEN BEGIN READ (Choix); READ(Ch∫
      oix); END;
      I2:=I2+1;
      IF I2>Table2[0].Index THEN BEGIN I2:=1; I1:=∫
      I1+1; END;
    END;
  END;

BEGIN (*debut du programme*)
  Plafond:=Dim*(Dim DIV 2);
  PAGE(OUTPUT);
  Initfichier;
  IF Erreur=FALSE THEN REPEAT
    WRITELN;
    WRITELN('choisissez: Ajouter une fiche, Recher∫
    cher, Lister, Quitter :');
    WRITELN('(ou "à": ajout aleatoire)');
    READ(Choix); WRITELN;
    CASE Choix OF
    'A','à': Ajouter; (*'à': pour essai*)
      'L': Lister;
    'R': Rechercher;
    END;
  UNTIL Choix='Q';
  CLOSE(Fich);CLOSE(Fitab);
END.
```

# Micro-Informations

## Jean-Michel Gourévitch

Rafale de nouveautés chez Apple. Elles ont été dévoilées en août et feront leurs débuts européens à l'Apple Expo.

La première de ces innovations est baptisée **HyperCard**, après avoir été connue sous le nom de code **WildCard**. Que les lecteurs anglophones ne se laissent toutefois pas abuser par cette désignation : il ne s'agit pas d'une carte d'extension, mais bel et bien d'un programme. Le plus difficile est d'expliquer exactement à quoi il sert. Chez Apple, on le décrit seulement comme un 'hypermédia' ou langage de navigation sur les bases de données. C'est en effet la première caractéristique de cette application, qui constitue un 'Hypertexte'.

L'**hypertexte** est un programme permettant, en cliquant sur un dessin à l'écran d'obtenir des informations plus détaillées sur ce dessin, et ainsi de suite. La firme **Owl**, fut la première à dévoiler un produit de ce genre avec **Glue**. Elle fut donc la première à râler lorsque Apple dévoila son HyperCard. D'autant que le programme d'Apple sera distribué gratuitement, eh oui gratuitement, avec tous les Macintosh.

Certaines applications déjà réalisées laissent entrevoir les utilisations de cet outil. L'une est un manuel de réparation de bicyclettes. Lorsque l'on clique sur une roue, on voit s'ouvrir une nouvelle fenêtre détaillant la pièce sur laquelle on a cliqué, et ainsi de suite. L'autre est le manuel d'utilisation de l'HyperCard, qui se présente à l'écran comme un bloc note illustré. On clique sur le titre d'un sujet, ou sur une information qu'on souhaite détailler et la page se tourne pour afficher la leçon. Très joli. Rien d'étonnant, si cet outil semble voué à la recherche des millions d'informations stockées, par exemple, sur un CD Rom laser.

Œuvre de **Bill Atkinson**, le père de MacPaint. Apple Hypercard est en fait un programme de gestion de documents utilisant des fiches (cards) et des piles (stacks). La programmation consiste à établir des relations entre cartes et piles grâce à des 'boutons' (des zones sur lesquelles on clique à l'écran). Ces boutons sont créés simplement grâce à un menu contenant des outils, il reste ensuite à définir l'action provoquée par un clic de souris : l'ouverture d'une autre fiche, ou grâce à un script (séquences de commandes d'un langage baptisé **HyperTalk** et comprenant une cinquantaine d'instructions) la mise en oeuvre d'une série d'actions (les programmeurs noteront qu'on peut même ouvrir alors une application, ou déboucher sur un programme écrit en **MPW**, le langage de programmation d'Apple).

HyperCard a déjà suscité tout un foisonnement de mini-applications baptisées *stackware*. Comme, par exemple, un guide touristique interactif. Bref, beaucoup de bénéfices pour pas grand chose, puisque HyperCard sera distribué, rappelons le, gratuitement. Le seul vrai problème est que ce programme consomme à lui seul 360Ko d'une disquette, et que les fichiers qui incorporent beaucoup de graphiques sont encore plus voraces. Avec HyperCard, le standard minimum du Mac vient de passer à une configuration incluant au moins un disque dur de 20 Mégas. Faute de quoi, on ne peut vraiment utiliser cette application géniale et dont on n'a pas fini de parler.

Et ce n'est pas fini, car l'autre nouveauté d'Apple, le **MultiFinder** est lui particulièrement exigeant en mémoire vive. Connu sous le nom de **Juggler**, ce MultiFinder est incorporé à la dernière version du Finder et repose sur les programmes écrits par **Andy Hertzfeld** sous le nom de **Servant**. Il s'agit d'un commutateur d'applications intégré au Bureau du Macintosh.

À la différence du Switcher, après avoir cliqué deux fois sur le MultiFinder, les applications deviennent simplement des fenêtres posées sur le bureau que l'on peut rétrécir par leur case de contrôle de taille. En dessous c'est toujours le bureau. On passe d'une application à l'autre en cliquant sur sa fenêtre et son menu vient alors prendre place dans la barre des menus du Macintosh. Pratique, pour échanger des documents entre les programmes et lorsque les auteurs des programmes l'auront prévu, on pourra réaliser une impression en tâche de fonds. On peut ouvrir simultanément 30 applications.

À condition de disposer de suffisamment de mémoire vive. Car, et c'est là que le bât blesse, il faut au minimum 2 Mégas de mémoire pour utiliser vraiment le MultiFinder (1 Méga est un peu juste, et 512Ko sont définitivement insuffisants). Une seule solution : acheter des extensions de mémoire. Il y a de la râlerie dans l'air...

Autre nouveauté en provenance d'Apple : une imprimante baptisée **ImageWriter II LQ**. La LQ dispose d'une résolution de 216 x 216 points par pouce. Soit le double de l'ancienne **Image Writer**, et à peine moins que la LaserWriter (300 x 300 points par pouce). Cette imprimante matricielle de très haute qualité

qui ne dispose cependant pas du langage de description de page **PostScript** (comme la Laser) est vendue aux alentours de 13 000 Francs HT. Dans le même temps, la Laser a baissé de plus de 20% passant de 49 900 F à 39 900 F HT. L'offre d'Apple se resserre donc. On attend pour les prochains mois une imprimante laser de très haut de gamme qui pourrait frôler les 80 000 Francs, et offrirait une résolution supérieure à 400 points par pouce, se rapprochant encore davantage des machines d'imprimerie professionnelle.

En bas de gamme de laser, on attendait une machine Apple. Surprise, c'est **General Computer** qui l'a sortie. Baptisée **Personal Laser Printer**, cette imprimante vendue 2 600 dollars aux États-Unis pourrait arriver en France aux alentours de 26 000 Francs. Elle ne possède pas le langage de description de page PostScript (ce qui exclut actuellement de pouvoir l'utiliser avec le programme **Illustrator** d'**Adobe**). On peut cependant imprimer des documents réalisés avec **PageMaker**, qui s'est véritablement imposé comme un standard dans l'édition électronique, dont nous allons parler plus loin.

En attendant, on travaille bien évidemment chez Apple à toute une salve de nouveautés. On a ainsi entendu parler d'un **Mac SE** qui pourrait inclure un écran couleurs, voire la carte du **Mac II** avec un seul connecteur d'extensions, d'un SE, qui serait vendu, à l'inverse, sans écran, etc. Il est sûr qu'une équipe travaille sur un micro-ordinateur équipé d'un processeur 68030.

On se rapproche peu à peu du pari de John Sculley, qui voudrait offrir un micro-ordinateur offrant une puissance de 100 Mips (Millions d'instructions par seconde), c'est-à-dire la puissance des volumineuses unités centrales d'aujourd'hui, et ce avant l'an 2000.

Côté programmes, la démarche d'Apple de créer une filiale de fabrication de logiciels (dont elle pourrait toutefois devenir actionnaire minoritaire) a suscité quelques remous aux États-Unis. Il s'agit en fait de contrebalancer la position dominante de **MicroSoft**.

En Europe, une unité de programmes stratégiques se charge d'aider les développeurs de logiciels. On peut déjà discerner les créneaux qu'Apple va tenter de favoriser. En premier lieu, bien évidemment, la publication électronique, en second lieu, les langages de programmation. Et puis toutes les applications tirant parti de la puissance et des caractéristiques exclusives du Mac II.

Ainsi, il faut s'attendre à voir encore davantage de logiciels de CAO et d'architecture. C'est que, comme le remarquait récemment une revue de CAO américaine, l'Europe occupe dans ce domaine une position privilégiée. Bien que le nombre d'architectes rapporté aux habitants soit plus faible de ce côté-ci de l'Atlantique. Enfin, seront privilégiés tous les programmes scientifiques, permettant d'implanter le Macintosh dans des bureaux d'études, ou au contrôle de production en usine. On n'a pas fini d'en voir. Et de toutes les couleurs, grâce à la marée montante des écrans polychromes pour le Mac II...

## Traitements de texte à gogo

Tandis que les utilisateurs de **Writer Plus** continuent à se désoler des bugs sauvages d'un programme par ailleurs génial, **Word** s'est imposé sur le marché. Peut être pas pour longtemps. Car voici que **WordPerfect**, le géant du traitement de texte sur IBM PC s'apprête à sortir une version pour le Mac qui va faire quelque bruit. Avec des macro-commandes, un vérificateur orthographique et un dictionnaire de synonymes, le multi-colonnage à l'écran, la césure automatique, la génération de notes de bas de page, tables des matières, d'index et de traitements d'idées. Et surtout la possibilité d'échanger des fichiers avec ceux créés sur les IBM, les Vax de Digital Equipment et les Data General.

En France, **Talor** déjà auteur d'un traitement de texte pour l'IBM PC a développé **Textor 4** pour le Macintosh capable, lui aussi d'échanger des fichiers avec le monde MS DOS. Textor vise le marché des techniciens avec un éditeur de formules mathématiques et un tableur.

Autre traitement de texte vedette outre Atlantique : **FullWrite**, un logiciel avec multi-colonnages générations de "post it notes" qu'on peut 'coller' sur un document et système de dessin incorporé, avec possibilité d'habillage automatique d'images irrégulières par un un texte...

## Publication électronique : le grand débat

Du côté de la publication électronique, on s'agite aussi beaucoup. **Aldus** a sorti la version 2.0 de **PageMaker** et travaille déjà à une version 3.0 encore perfectionnée. **LettraSet** a sorti une version 4.0 de **Ready Set Go!**. On y dispose notamment d'un bureau à la présentation plus claire avec une main pour déplacer la page, le tracé de lignes diagonales, une palette de filets disponible par menus, un espacement contrôlé entre les mots, la spécification possible des césures avec un dictionnaire d'exceptions, un glossaire, un vérificateur orthographique, l'habillage automatique de dessins, etc. Quant à **RagTime**, importé par **ItalSoft**, il s'offre déjà une version 2 avec habillage des réserves, crénage, 40 trames de fond, une grille de construction, un traitement de texte avec césure automatique algorithmique, un dictionnaire d'exceptions personnalisables, etc.

Au sommet des programmes d'édition, le match s'annonce passionnant entre **PageMaker** et **XPress** de **Quark**, vendu chez nous par P-Ingénierie. Disons simplement en résumé, que **XPress** grâce à sa création obligatoire de "réserves" pour le texte ou l'image peut être plus facilement manipulé par quelqu'un n'ayant aucune notion de mise en page, mais pour qui l'on a pré-défini des gabarits. En revanche, **PageMaker** qui accepte n'importe où des textes ou des images, sans que l'on ait besoin de le spécifier se révèle souple et particulièrement rentable aux mains d'utilisateurs imaginatifs.

L'arrivée de **Scoop** pourrait encore agiter ce nouveau secteur en plein développement et où le Macintosh tient toujours le haut du pavé.

## L'avenir ?

L'année 1988 se révélera cruciale pour Apple. Si la firme de Cupertino sait bien utiliser ses arguments et étendre ses positions dans des domaines où elle est en pointe, elle peut profiter de l'instabilité instaurée par la rupture de standard du monde IBM. Il ne serait pas étonnant d'assister dans ce débat à une alliance entre **DEC** (le spécialiste numéro 1 des mini-ordinateurs) et Apple pour faire la nique au géant IBM. On constate déjà un grand nombre de connections réalisées entre des Mac et des VAX de DEC. L'apparition de cartes permettant de relier les Mac aux réseaux **Ethernet** (avec notamment le logiciel **3 COM Plus**) est un signe important dans ce sens.

Quant à AppleTalk, le réseau de connections d'Apple, il est aujourd'hui tout simplement le plus répandu au monde reliant entre eux quelque 450 000 Macintosh. L'avenir du Mac est donc dans les réseaux. On s'éloigne à grands pas de la machine des "travailleurs du savoir" vantée jadis par Steve Jobs.

Et si c'était la clé du succès ?



# Sur CalvaCom

Dans cette nouvelle rubrique, nous vous présentons une sélection des questions/réponses échangées sur la messagerie de CalvaCom. Notre boîte à lettres : emp11.

7) Bonjour. *AR* - 14 l.
De: Cédric NEROT (CN10) - 02 sep 87 22h26
Bonjour !
Enfin je trouve 5 minutes pour faire exploser ma joie ! Pom's est sur Calva !!! Nous passerons rapidement sur tout le bien que je pense de vous (L'heure de connexion ici bas n'est malheureusement pas donnée !) pour arriver à ma question...
Envisagez vous de télécharger les programmes de la revue via Calva, avec, si besoin est, dédommagement ? Il semble en effet bien plus pratique, et bien plus rapide (pour ne pas dire : plus sur, avec les Postes niçoises) de récupérer un texte de cette façon que par une disquette soumise aux voies de fait postales.. Qu'en pensez vous ?

Amicalement & admirativement votre...

*Merci, vous nous faites rosir... Nous pensons dans un premier temps mettre à disposition sur bibliothèque CalvaCom des programmes mais nous réfléchissons également à un projet plus large : la chose est à l'étude.*

4) Questions naives - 43 l.
De: Patrick BARCZEWSKI (PB46) - 04 aou 87 11h06

Bonjour à tous, et merci d'avance du temps que vous voudrez bien consacrer aux questions suivantes qui vous sembleront sans doute bien candides.

1/UPGRADE
Abonné à Pom's, je dispose des programmes suivants :
1/ Minitel (disquette commandée avec votre numéro 27)
2/Interpom's (disquette commandée avec votre numéro 28)
Je souhaite les mettre au niveau des versions décrites dans votre numéro 31, et vous commander le programme CLV_POM'S.
Un tarif spécial est-il prévu ?

2/AFFICHAGE 80 COLONNES
Le serveur envoie en ligne au minitel IB la commande qui le fait basculer en mode 80 colonnes. J'ai des problèmes pour restituer localement un fichier sur lequel figure un tel passage.
En d'autres termes, en mode consultation de votre programme minitel, quelle est la séquence de touche qui permet de reproduire localement sur le IB cette commande du serveur?

3/FILTRAGE DES CARACTERES ACCENTUES
Votre programme minitel, mode préparation de texte, avec lequel ce message est passé sur Calvacom après préparation sous Macwrite, semble, comme vous pouvez le constater, avoir des difficultés à filtrer les caractères accentués. Comment y remédier?

Encore bravo pour ces programmes de comm vraiment utiles.

Cordialement,

Patrick Barczewski (PB46)

*1/UPGRADE*
*Pour une question de droits d'auteur (modif IBM obligent), nous ne pouvons envisager de tarifs spéciaux pour InterPom's, même pour les fidèles...*
*La nouvelle version Mac du programme Minitel du n° 27 est sur la disquette 31, en votre possession si vous êtes abonné.*

*2/AFFICHAGE 80 COLONNES*
*Le Minitel passe en mode téléinformatique français par la séquence : FNCT T suivi de F.*
*NB: Pour l'enregistrement de CalvaCom, CLV_POMS est pratique.*

*3/FILTRAGE DES CARACTERES ACCENTUES*
*Le programme du 27 était conçu pour les serveurs Vidéotex ce qui explique vos difficultés (qui dépendent du type de Minitel).*
*CLV_POMS résout le problème...*

6) questions sur clv-poms - 8 l.
De: Gerard MARTZ (GM46) - 31 jul 87 15h16

Le programme clv-poms permet-il de réceptionner également des fichiers binaires issus des bibliothèques ?
Ce point n'est pas clairement expose dans l'article de pom's 31 !

Merci de me répondre en bal gm46

*Si vous avez un Apple //, seuls les fichiers TEXT ou EXE peuvent être importés, mais la plupart des programmes en langage machine ou en Basic sont disponibles sous la forme d'un fichier TEXT EXECutable. Pour le Mac, vous importerez en priorité BinHex, fichier Basic qui créera l'application BinHex4. Avec ce programme et CLV_Poms, tous les fichiers sont récupérables (PackIt 3 également en bib vous sera utile pour certains).*

1) TRANSFERT FICHIERS TXT EN RAM - 25 l.
De: Francois MULLER (FM17) - 06 aou 87 09h53

Chers Amis
Merci d'abord d'avoir ouvert une BAL pour communiquer avec vous.
J'ai un peu avancé depuis mon dernier courrier (transfert de fichiers TEXT en RAM)...
J'ai transformé le programme COPYBASFILES de Francois Dreyfuss paru dans le numéro 28 de POM'S.
Pour transférer des fichiers TXT en RAM il faut faire les Modfs suivantes:

```
  5 DIM L$(40), O$(40)
155 P$=RIGHT$(L$(I),12)
156 O$(I)=LEFT$(P$,5)
164 " BAS " = " TXT "
595 PRINT "CREATE"R$;L$(I);",T TXT"
600 PRINT "BLOAD";MM$;L$(I);",T TXT,A$ A00"
700 PRINT "BSAVE"R$;L$(I);",T TXT,A$ A00,L ";O$(I)
```

Tous ce passe bien mais rapidement j'ai le message - DIRECTORY FULL -
Exemple: je mets 12 fichiers TXT en RAM et j'ai ce message...

Pourtant CAT/RAM me donne
- BLOCKS FREE: 93
- BLOCKS USED: 34 Où est le problème ?

PS : que signifie le T15 a la ligne 90 ?

Merci d'avance pour votre réponse

*Normal : Le directory est effectivement saturé (pas le disque) car il n'est composé que d'un bloc au lieu de quatre pour les lecteurs classiques.*
*Chaque bloc peut contenir 13 fichiers sauf le premier qui contient des informations sur le support et sur le catalogue lui-même : il ne peut en recevoir que 12.*
*Sur votre lecteur 5,25 pouces, le directory est saturé avec 51 fichiers :*
*12 + 13 + 13 + 13 = 51.*

*La solution réside dans l'utilisation de sous-directory qui, eux, ne sont pas limités en taille :*
*Pour en créer un : PRINT D$"CREATE /RAM/SOUS.CAT,TDIR"*

*T15 signifie type directory, c'est équivalent à T$F et TDIR.*

4) Pom's - 4 l.
De: David BENSIMON (DB25) - 14 sep 87 12h11

Votre revue est super super super !!
Bravo et continuez avec cette qualité !

David Bensimon.

*Merci, nous en restons sans voix...*

4) Cordon MAC + MINITEL - 6 l.
De: NUMERA (PN19) - 08 jul 87 21h07

Bonjour
Philippe C. m'a dit que vous pourriez me procurer un cordon MAC + /MINITEL. Quel en est le prix et serait il possible d'en avoir un d'une longueur de 5 m environ si cette longueur n'est pas critique.
En attendant impatiemment de vos nouvelles...
Je vous remercie d'avance..-

*Nous pouvons effectivement effectuer cette liaison qui vous coûtera 300,00 F Franco.*

---

4) prog clv + pb export - 11 l.
De: Olivier FAGES (OF17) - 04 sep 87 00h27

Bonjour, Je suis fidèle lecteur et utilisateur des PROGS de Pom's. En particulier, CLV et MINBAS qui m'aident à découvrir CALVACOM. Mais j'ai des pbs :

1-Avec CLV, j'obtiens des fchiers en inverse et non exploitables avec AppleWorks, utilisables avec AppleWriter (mais après CONVERT!!). Y a-t-il une solution à ce pb.

2-CALVACOM incite à EXporter mais le 75 bauds est limitatif. Avec un minitel 1B, votre génie, 1 Apple série, ne peut-on pas faire plus vite (SANS modem) ou est-ce une limitation du serveur spécial minitel je n'y connais pas grand chose et compte sur vous ou un forum.

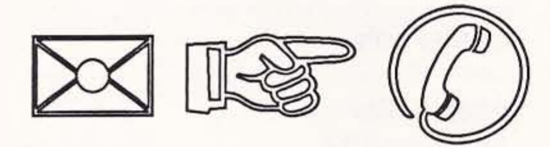
olivier /of 17

*Nous n'avons pas vos difficultés avec notre AppleWorks qui semble indifférent au 'poids fort' des octets composants le fichier TEXT. Nous avons toutefois apporté une modification à CLV dans sa version 1.01, mais n'ayant pas le problème, nous ne savons pas s'il est résolu...*

*2 solutions :*
*- nous renvoyer votre disquette pour échange,*
*- nous téléphoner si vous avez InterPom's pour télécharger la nouvelle version.*

*CalvaCom ne gère pas le retournement du Minitel pour qu'il émette à 1200 bauds, donc, pas de solution immédiate.*

---



---



# Mots croisés

Pour cette première grille de mots croisés de Pom's, nous vous proposons un petit concours : soyez le premier à nous envoyer la solution et vous bénéficierez d'un avoir de 200,00 F sur les produits Pom's, pourquoi pas Ordico ?...

Les possesseurs de la disquette d'accompagnement pourront résoudre le problème sur leur Macintosh grâce à l'application – écrite en Turbo-Pascal par Roland Jost – présentée dans ces pages.

Problème 32 par Joelle Piard

Effacer | Sauver | grille n° 1
Imprimer | Quitter

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | | | | | | | | |
| 2 | | | | | | | | | | |
| 3 | | | | | | | | ■ | | |
| 4 | | | | | ■ | | | | | |
| 5 | | | | | | ■ | | | ■ | |
| 6 | | | | ■ | | | | | | ■ |
| 7 | | | | | | | | | | |
| 8 | | | | | | | | | ■ | |
| 9 | | | | | | | ■ | | | |
| 10 | | | | ■ | | | | | | |

Horizontalement

1- Pas très bas 2- Reprises
3- Parasites - Tout un sac, c'est pas épais
4- Sur un arbre, mais pas un fruit - Pour les babies
5- Pour un essai, c'est raté - Deux pour les Zazous
6- Pour le coureur de fonds - Deux pour un parfum
7- Sans ordre
8- Roche rouge
9- Plainte mélancolique - Mauvais pour la vigne, bon pour bébé
10- Possessif - Distant

Verticalement

1- Ne pleurent pas entre les rails 2- Française 3- Toujours plus proches
4- Courant - On peut y trouver le 8 horizontal
5- Abréviation couronnée - Amère 6- Possessif - Pour tailler le 8 horizontal
7- Subissant 8- Deux à Paris - Ne reconnaîtra plus
9- Fut en Asie Mineure - Une pomme grand sport - Oui
10- Sur certaines tables - Poursuit sans courir

| Disquettes Apple ][ | | | Macintosh | | | Accompagnement | | |
|---|---|---|---|---|---|---|---|---|
| CLV_Pom's - 140Ko ou 800Ko | 200,00 F | ☐ | CLV_Pom's | 200,00 F | ☐ | A ][ 140Ko 4 | 60,00 F | ☐ |
| Éditeur Vidéotex - 140Ko | 200,00 F | ☐ | Mac Raccourci | 200,00 F | ☐ | A ][ 140Ko 5 | 60,00 F | ☐ |
| E.P.E. 5.1 - 140Ko | 200,00 F | ☐ | MacAstuces | 200,00 F | ☐ | A ][ 140Ko 6 | 60,00 F | ☐ |
| E.P.E. 5.1 - 800Ko | 200,00 F | ☐ | Mac 'A' | 80,00 F | ☐ | A ][ 140Ko 7 | 60,00 F | ☐ |
| Max (moniteur étendu) - 140Ko | 150,00 F | ☐ | Mac 'B' | 80,00 F | ☐ | A ][ 140Ko 8 | 60,00 F | ☐ |
| BananaSoft - 140Ko | 200,00 F | ☐ | Mac 'C' | 80,00 F | ☐ | A ][ 140Ko 9 | 60,00 F | ☐ |
| Pascal - 140Ko | 80,00 F | ☐ | Mac 'D' | 80,00 F | ☐ | A ][ 140Ko 10 | 60,00 F | ☐ |
| Max (moniteur étendu) - 140Ko | 150,00 F | ☐ | Mac 'E' | 80,00 F | ☐ | A ][ 140Ko 11 | 60,00 F | ☐ |
| Dominos - 140Ko | 80,00 F | ☐ | Mac 'F' | 80,00 F | ☐ | A ][ 140Ko 12 | 60,00 F | ☐ |
| COGO - 140Ko | 200,00 F | ☐ | Mac 'G' | 80,00 F | ☐ | A ][ 140Ko 13 | 60,00 F | ☐ |
| Ludologic - 140Ko | 80,00 F | ☐ | Mac 'H' | 80,00 F | ☐ | A ][ 140Ko 14 | 60,00 F | ☐ |
| Ordico - 140Ko | 200,00 F | ☐ | Mac 'I' | 80,00 F | ☐ | A ][ 140Ko 15 | 60,00 F | ☐ |
| | | | | | | A ][ 140Ko 16 | 60,00 F | ☐ |
| Recueils Pom's | | | Mac 17 | 150,00 F | ☐ | A ][ 140Ko 17 | 60,00 F | ☐ |
| | | | Mac 18 | 80,00 F | ☐ | A ][ 140Ko 18 | 60,00 F | ☐ |
| Numéro 1 (Revues 1 à 4) | 140,00 F | ☐ | Mac 19 | 80,00 F | ☐ | A ][ 140Ko 19 | 60,00 F | ☐ |
| Disquettes 1 à 4 | 200,00 F | ☐ | Mac 20 | 80,00 F | ☐ | A ][ 140Ko 20 | 60,00 F | ☐ |
| Numéro 2 (Revues 5 à 8) | 140,00 F | ☐ | Mac 21 | 80,00 F | ☐ | A ][ 140Ko 21 | 60,00 F | ☐ |
| Disquettes 5 à 8 | 200,00 F | ☐ | Mac 22 | 80,00 F | ☐ | A ][ 140Ko 22 | 60,00 F | ☐ |
| Numéro 3 (Revues 9 à 12) | 140,00 F | ☐ | Mac 23 | 80,00 F | ☐ | A ][ 140Ko 23 | 60,00 F | ☐ |
| Disquettes 9 à 12 | 200,00 F | ☐ | Mac 24 | 80,00 F | ☐ | A ][ 140Ko 24 | 60,00 F | ☐ |
| | | | Mac 25 | 80,00 F | ☐ | A ][ 140Ko 25 | 60,00 F | ☐ |
| ______ reliures toilées | | | Mac 26 | 80,00 F | ☐ | A ][ 140Ko 26 | 60,00 F | ☐ |
| pour 6 numéros, soit un an | 60,00 F | ☐ | Mac 27 | 80,00 F | ☐ | A ][ 140Ko 27 | 60,00 F | ☐ |
| | | | Mac 28 | 80,00 F | ☐ | A ][ 140Ko28 | 60,00 F | ☐ |
| Apple ][ 800Ko 3'5 numéro 29 | 80,00 F | ☐ | Mac 29 | 80,00 F | ☐ | A ][ 140Ko 29 | 60,00 F | ☐ |
| Apple ][ 800Ko 3'5 numéro 30 | 80,00 F | ☐ | Mac 30 | 80,00 F | ☐ | A ][ 140Ko 30 | 60,00 F | ☐ |
| Apple ][ 800Ko 3'5 numéro 31 | 80,00 F | ☐ | Mac 31 | 80,00 F | ☐ | A ][ 140Ko 31 | 60,00 F | ☐ |
| Apple ][ 800Ko 3'5 numéro 32 | 80,00 F | ☐ | Mac 32 | 80,00 F | ☐ | A ][ 140Ko 32 | 60,00 F | ☐ |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Revue n° 8 | 35,00 F | ☐ | Revue n° 10 | 40,00 F | ☐ | Revue n° 11 | 40,00 F | ☐ |
| Revue n° 12 | 40,00 F | ☐ | Revue n° 13 | 40,00 F | ☐ | Revue n° 14 | 40,00 F | ☐ |
| Revue n° 15 | 40,00 F | ☐ | Revue n° 16 | 40,00 F | ☐ | Revue n° 17 | 40,00 F | ☐ |
| Revue n° 18 | 40,00 F | ☐ | Revue n° 19 | 40,00 F | ☐ | Revue n° 20 | 40,00 F | ☐ |
| Revue n° 21 | 40,00 F | ☐ | Revue n° 22 | 40,00 F | ☐ | Revue n° 23 | 40,00 F | ☐ |
| Revue n° 24 | 40,00 F | ☐ | Revue n° 25 | 40,00 F | ☐ | Revue n° 26 | 40,00 F | ☐ |
| Revue n° 27 | 45,00 F | ☐ | Revue n° 28 | 45,00 F | ☐ | Revue n° 29 | 45,00 F | ☐ |
| Revue n° 30 | 45,00 F | ☐ | Revue n° 31 | 45,00 F | ☐ | Revue n° 32 | 45,00 F | ☐ |

## Abonnements pour six numéros à partir du ______, à :

| | | |
|---|---|---|
| la revue seule | 225,00 F | ☐ |
| la revue et les disquettes Apple ][ 140Ko – 5' 1/4 | 525,00 F | ☐ |
| la revue et les disquettes Apple ][ 800Ko – 3' 1/2 | 625,00 F | ☐ |
| la revue et les disquettes Macintosh | 625,00 F | ☐ |
| la revue, les disquettes Apple ][ 140Ko – 5' 1/4 et les disquettes Mac | 925,00 F | ☐ |
| la revue, les disquettes Apple ][ 800Ko – 3' 1/2 et les disquettes Mac | 1025,00 F | ☐ |

Envoyez ce bon et votre règlement à : **Éditions MEV – 12, rue d'Anjou – 78000 Versailles**

**Nom :** ______________________________

**Adresse :** ______________________________

**Règlement par :** Carte Bleue/VISA ☐ Chèque bancaire ☐ Chèque postal ☐ Mandat ☐

numéro de la carte _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ date d'expiration _ _ _ _ _ _

Montant _ _ _ _ , _ _ F Signature :

* pour les envois par avion, ajoutez 15 F par numéro et/ou par disquette soit, par exemple, 90 F pour un abonnement avec ou sans disquettes.

# Câble-interface de communication Apple/Minitel

Cette liaison – décrite dans les numéros 27 et 28 de Pom's – est indispensable pour faire fonctionner les programmes suivants :

- **MinBas** pour Apple ][+, //e, //e+, //c et //GS* : programme permettant l'enregistrement des écrans Minitel, la restitution à loisir hors réseau, le stockage et/ou l'impression de copies d'écran du Minitel, et aussi l'envoi de textes ou messages sur un serveur, ces messages étant préparés à l'avance avec n'importe quel programme de traitement de textes. Programme publié dans le numéro 27 de Pom's.
- **Minitel/1** pour Macintosh : programme identique à MinBas pour Apple ][, avec en plus un mini-éditeur de texte pour la préparation et le stockage des messages sans sortir du programme. Programme publié dans le numéro 27 de Pom's.
- **InterPom's** pour Apple ][+, //e, //e+, //c et //GS* : programme de téléchargement entre Apple ][ et/ou Apple ][ et Macintosh. Ce logiciel autorise la transmission de n'importe quel type de fichier (système, texte, binaire, Basic...) en utilisant le Modem du Minitel. Programme publié dans le numéro 28 de Pom's.
- **InterPom's** pour Macintosh : même programme, mais version Macintosh. Programme publié dans le numéro 28 de Pom's.
- **TPom's** pour Apple ][+, //e, //e+, //c et //GS** : programme de récupération de l'annuaire électronique sous la forme de fichiers texte. Voir page 53 du numéro 30.
- **TPom's** pour Macintosh : Identique à la version Apple ][. Voir page 42 du numéro 30.

* ce programme est prévu pour fonctionner avec une carte Super Série Apple ; le port série intégré de l'Apple IIGS ne convient pas. Toutefois, si vous désirez utiliser le port intégré afin d'y connecter le Minitel pour d'autres applications, commandez un câble pour Macintosh Plus.

** sur un Apple IIGS, TPom's fonctionne indifféremment avec le port série intégré ou la carte Super Série Apple. Pour connecter le port intégré du IIGS, utilisez un câble pour Macintosh Plus.

| | | | | |
|---|---|---|---|---|
| **Je désire recevoir :** câble Minitel/Apple ][+, //e, //e+, IIGS avec SSC | ____ | à | 225,00 F | ________ |
| câble Minitel/Apple //c | ____ | à | 225,00 F | ________ |
| câble Minitel/Macintosh 128 & 512Ko | ____ | à | 225,00F | ________ |
| câble Minitel/Macintosh Plus, IIGS port intégré | ____ | à | 225,00 F | ________ |
| câble Minitel/IBM PC™ | ____ | à | 225,00 F | ________ |
| câble de liaison locale Apple II/Mac/IBM™ * | ____ | à | 225,00 F | ________ |

*(préciser le type des deux machines à relier : Mac 512, Mac Plus, Apple //e //c, IIGS, IBM PC™ )

* pour les envois par avion, ajoutez 15 F par câble.

---

Envoyez ce bon et votre règlement à : **Éditions MEV – 12, rue d'Anjou – 78000 Versailles**

**Nom :** ______________________________

**Adresse :** ______________________________

**Règlement par :** Carte Bleue/VISA ☐ Chèque bancaire ☐ Chèque postal ☐ Mandat ☐

numéro de la carte _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ date d'expiration _ _ _ _ _ _

Montant _ _ _ _ , _ _ F Signature :

---

# Programme de transmission InterPom's

Ce programme décrit dans le numéro 31 vous donne la possibilité d'échanger à distance via Minitel et en local des fichiers de texte entre Apple //, Macintosh et IBM PC™. Entre deux machines de même type, tous les fichiers sont transférables : Applications, polices de caractères, accessoires de bureau, commandes, documents graphiques etc.

| | | | | |
|---|---|---|---|---|
| **Je désire recevoir :** InterPom's pour Apple //, disquette 140Ko, 5'25 | ____ | à | 450,00 F | ________ |
| InterPom's pour Apple //, disquette 800Ko, 3'25 | ____ | à | 450,00 F | ________ |
| InterPom's pour Macintosh, disquette 3'25 | ____ | à | 450,00 F | ________ |
| InterPom's pour IBM PC™ | ____ | à | 450,00 F | ________ |

* pour les envois par avion, ajoutez 15 F par disquette.

---

Envoyez ce bon et votre règlement à : **Éditions MEV – 12, rue d'Anjou – 78000 Versailles**

**Nom :** ______________________________

**Adresse :** ______________________________

**Règlement par :** Carte Bleue/VISA ☐ Chèque bancaire ☐ Chèque postal ☐ Mandat ☐

numéro de la carte _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ _ date d'expiration _ _ _ _ _ _

Montant _ _ _ _ , _ _ F Signature :

Dépôt légal : septembre 1987, n° d'impression 789441